



DEEP LEARNING FOR POINT CLOUD SEGMENTATION WITH APPLICATIONS TO SAWMILL INDUSTRY

Lappeenranta-Lahti University of Technology LUT

Master's Program in Computational Engineering, Master's Thesis

2022

Maksim Shchukin

Examiner: Assoc. Prof. Tuomas Eerola
 Prof. Lasse Lensu

ABSTRACT

Lappeenranta-Lahti University of Technology LUT
School of Engineering Science
Computational Engineering

Maksim Shchukin

Deep learning for point cloud segmentation with applications to sawmill industry

Master's thesis

2022

50 pages, 24 figures, 13 tables, 0 appendices

Examiners: Assoc. Prof. Tuomas Eerola and Prof. Lasse Lensu

Keywords: point cloud, machine vision, pattern recognition, sawmills, computer vision, deep learning

Nowadays, the integration of digital technologies in enterprises is happening everywhere. The sawmill industry is no exception. The various stages of production at sawmills are being improved through the use of information technology and machine learning. In particular, the use of a 3D digital log model is very beneficial as it allows to simulate internal knot distribution and select optimal sawing pattern. One of the main problems is that 3D models of logs obtained using laser scanners often contain noise and various extraneous objects. This hinders the correct analysis. The noise can be filtered from the log using segmentation algorithms. Some solutions to this problem exist, but they require manual parameter configuration. The goal of this thesis was to solve this problem using modern deep learning approaches. More specifically, the aims were to review existing solutions for point cloud segmentation, select the most appropriate one, train and evaluate it. After consideration, RandLA-Net was selected because it can process large point clouds in a reasonable amount of time, and also due to its ability to accurately segment complex geometric structures. The selected method accurately distinguished noise and logs. It also demonstrated the ability to segment previously unseen data that was gathered in a different environment.

ACKNOWLEDGEMENTS

I would like to thank my supervisors Heikki Kälviäinen, Lasse Lensu, Tuomas Eerola, and Fedor Zolotarev for their guidance during the work on this thesis. I am also grateful to my family and friends for their support.

Lappeenranta, December 9, 2022

Maksim Shchukin

LIST OF ABBREVIATIONS

2-D	Two Dimensional
3-D	Three Dimensional
CNN	Convolutional Neural Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
FCN	Fully Convolutional Network
FN	False Negative
FP	False Positive
GPU	Graphics Processing Unit
MLP	Multilayer Perceptron
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
TN	True Negative
TP	True Positive

CONTENTS

1	INTRODUCTION	7
1.1	Background	7
1.2	Objectives and delimitations	8
1.3	Structure of the thesis	9
2	SAWMILLING	10
2.1	Sawmill pipeline	10
2.2	Machine learning for sawmill industry	11
2.3	Point cloud processing for sawmilling	12
3	DEEP LEARNING	13
3.1	Background	13
3.2	Common components of neural networks	13
3.3	Neural network training	14
3.4	Multilayer perceptron	15
3.5	Convolutional neural network	15
4	POINT CLOUD SEGMENTATION	18
4.1	Background	18
4.2	Projection-based approaches	19
4.2.1	Multi-view representation	20
4.2.2	Spherical representation	20
4.3	Discretization-based approaches	21
4.3.1	Dense discretization representation	22
4.3.2	Sparse discretization representation	23
4.4	Point-based approaches	23
4.4.1	Shared multilayer perceptron methods	23
4.4.2	Point convolution methods	24
4.4.3	Recurrent neural network methods	24
4.4.4	Graph-based methods	26
4.5	Hybrid methods	26
5	3D POINT CLOUD SEMANTIC SEGMENTATION	28
5.1	Justification of choice	28
5.2	Architecture	28
5.2.1	Local Spatial Encoding	29
5.2.2	Attentive Pooling	30
5.2.3	Dilated Residual Block	30

	6
5.2.4 RandLA-Net	31
6 EXPERIMENTS	33
6.1 Data	33
6.2 Description of experiments	34
6.3 Evaluation criteria	36
6.4 Results	37
7 DISCUSSION	44
7.1 Current study	44
7.2 Future work	45
8 CONCLUSION	46
REFERENCES	47

1 INTRODUCTION

1.1 Background

About 75% of Finland's territory is covered with forests. The most common trees for industrial use are: birch, spruce and pine [1]. The country is one of the top 10 softwood exporters in the world. Products from the sawmill industry are one of Finland's most important exports. About 24,000 people are employed in this industry. There are more than 80 industrial sawmills in Finland and hundreds of smaller local mills. In 2018, the total export value of this industry was 1.8 billion euros. In 2021, about 11.5 million cubic meters of softwood were produced [2]. It is essential for the sawmill industry to perform process optimization, as it allows to increase the profitability of the business and improve product quality [3].

This work is related to research carried out within the DeepTimber and DigiSaw projects. The DigiSaw aims to improve the sawmilling industry through digitalization and the use of machine learning approaches [4]. The goal of DeepTimber is similar, it focuses on sawing optimization. Modern sawmills utilise a range of high-tech equipment such as laser scanners. Laser scanners are used to create a 3D point cloud from the log surface. It captures the shape and size of the log. The laser scanners work fast, especially in comparison with other alternatives like CT scanner. However, the data obtained by laser scanners often contain noise and outliers. The scanner can capture various objects such as metal rails, which need to be filtered out later [3]. In the earlier study [3], the approach for solving this problem was to perform point cloud filtering using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [5] algorithm layerwise along the z -axis [3]. This approach has its drawbacks, for example, the accuracy of noise detection is not always satisfactory, also in this specific application the DBSCAN requires manual parameter tuning to adapt to some data.

This Master's thesis considers noise filtering problem with more robust and accurate approach based on deep learning (i.e. based on artificial neural network). Deep learning is preferable for this work due to its ability to automatically learn relevant features, and deep learning is usually done in an end-to-end manner. More specifically, the goal is to perform filtering using a 3D point cloud semantic segmentation method. Semantic segmentation is the process of classifying individual points into predefined classes. There are two classes in this work, namely noise and log. An example of the segmentation process is demonstrated in Fig. 1.

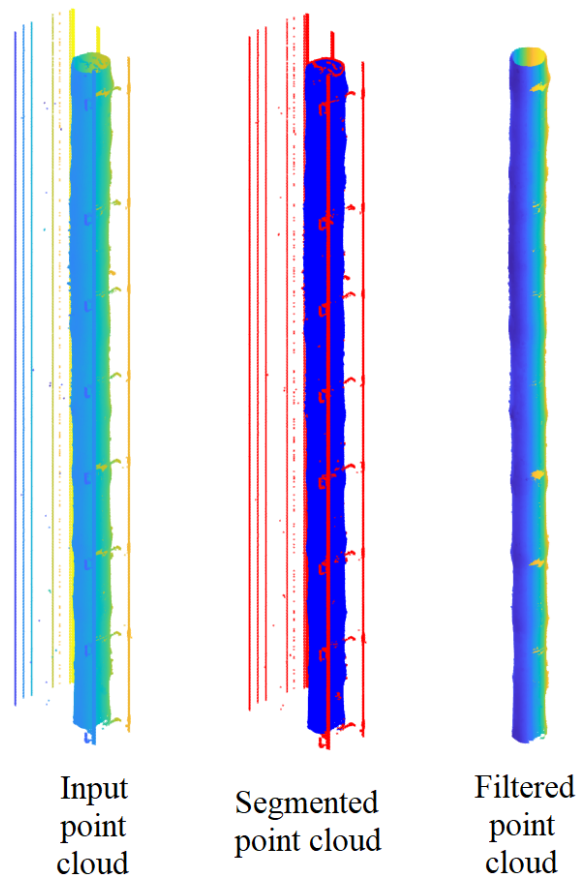


Figure 1. Segmentation process.

1.2 Objectives and delimitations

The main objective of this work is to perform point cloud segmentation using deep learning method to remove measurement noise.

The specific objectives of the master's thesis are as follows:

- Study the existing point cloud segmentation approaches and select the most appropriate one.
- Train the selected method for filtering 3D point clouds of wooden logs.
- Evaluate the method on real data collected in a sawmill.

Since the work continues previous research, the same delimitations apply [3]. The data contains only softwood point clouds.

1.3 Structure of the thesis

The standard stages of the sawmill process, as well as some of the machine learning algorithms used in it, are described in Chapter 2. Chapter 3 defines deep learning and introduces some of the modern methods and their components. Chapter 4 introduces point cloud segmentation task and overviews the existing algorithms for solving this problem. Chapter 5 describes the architecture of the selected point cloud segmentation algorithm, as well as its advantages. Chapter 6 describes the datasets used in this research, the experiments, and their results. The results of the research and potential future work are discussed in Chapter 7. Finally, the conclusions are given in Chapter 8.

2 SAWMILLING

2.1 Sawmill pipeline

A sawmill is the location where logs are processed into lumber. Modern sawmills are computerized, containing a large set of specialized equipment such as conveyors, planes and saws [6]. Sawmilling involves multiple different stages as shown in Fig. 2. They are listed as follows:

- **Felling, limbing and scaling.** The process begins by cutting down trees. After that the branches are removed from the trunks, this process is called limbing. Next, the trees are cut to length. Then the logs being scaled in order to evaluate wood volume. The next step is the transportation of the logs to the industrial facilities by rail or timber truck [6, 7].
- **Sorting, debarking and metal removal.** After delivery to the sawmill, the logs are sorted according to their type, size and later use. Next, the debarking process takes place. During this process, bark and various impurities are removed from the log. The next stage is the metal removal. At this stage the log is passed through a metal detector to check if it contains any metal particles. If it contains, then the log is removed from the conveyor and cleaned of metal. These last two stages are essential to extend the service life of the saws that are used in the next sawing stage [6, 8].
- **Sawing and green sorting.** The next stage is sawing. This is the process of cutting logs into planks by using circular and band saws. There are several sawing patterns. Three most frequently used are: flat sawing, quarter sawing and rift sawing. The log is analyzed for the optimal type of sawing selection. The selected cutting pattern should maximize value acquired after selling lumber (minimize wasted material). The resulted lumber have different sizes and thickness so it needs to be sorted. Usually, the wood is sawed before drying so the process is called green sorting [6,9].
- **Drying.** Drying is the next important stage. This stage is carried out to bring the moisture content of the wood to the minimum amount required for sale on the market. Usually, this step is performed in drying kilns in a batches. Dryers are filled with boards of the same thickness. The drying time depends on the thickness of the boards, type of the wood and the size of the kiln [10]. Today there are many wood drying technologies: conventional, dehumidification, solar, vacuum and radio frequency [6, 11]. However, the wood is not always dried, for some purposes it is used green [6].
- **Grading, packaging and delivery.** After drying, the lumber is measured and sorted by length. Then, based on the size, type of wood, the presence of defects and knots,

the price is estimated. The finished lumber products are being packaged and then transported to the customer. Depending on the customer's request, sometimes the lumber is additionally processed before delivery [6].

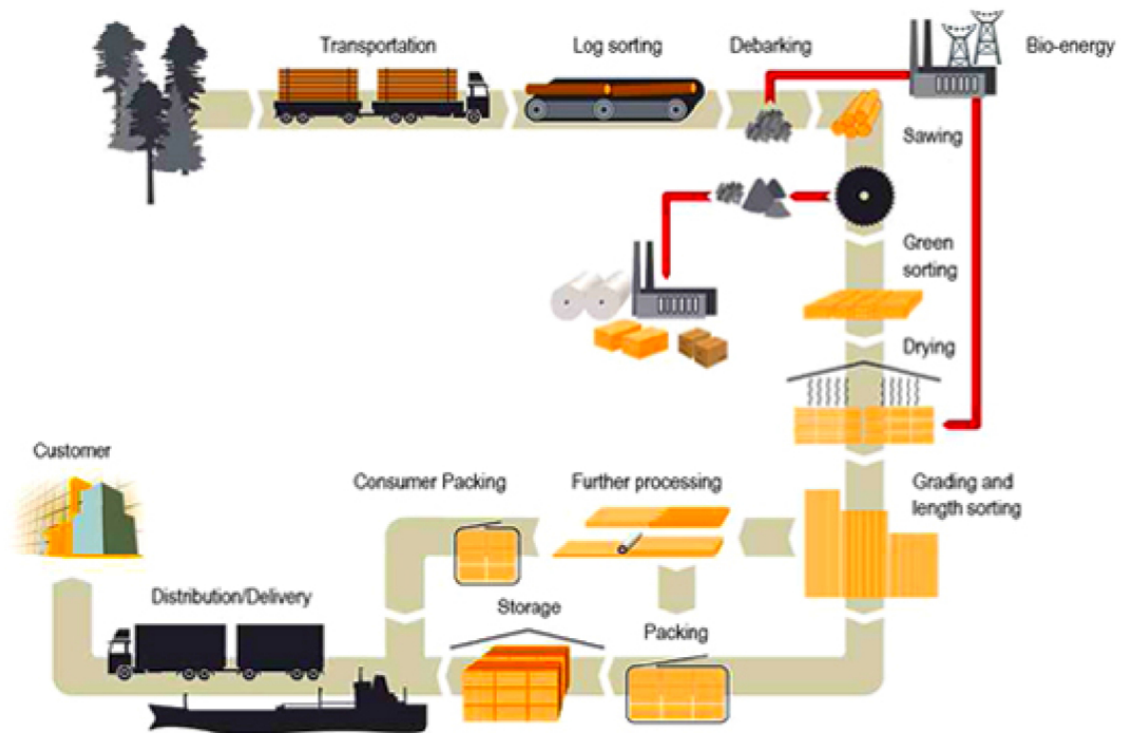


Figure 2. Sawmill process [6].

Also, the modern sawmills focus on minimizing waste and improving energy efficiency. Shavings, bark and sawdust are often burned to heat wood-drying kilns as bio-energy. The sawdust is also processed into particle boards. Some pieces of wood that are not suitable for making lumber are processed into chips and then sold to paper mills [6].

2.2 Machine learning for sawmill industry

For at least 50 years, the sawmill industry has been using various modeling and optimization approaches. The targets of optimization include: efficient land use, optimal construction of roads between forest and sawmill [12]. The machine learning models are used to address wood allocation planning problem, cutting pattern selection and sawing simulation [9].

There are various sawing simulators which use virtual log representation (e.g. 3D scan of log) for the prediction of the best type of cutting pattern. Some simulators consider the type of the sawing equipment and its configurations [9]. To make the modeling of virtual logs less computationally challenging neural networks, genetic programming, radial basis functions, Bayesian networks, and regression models have been proposed [9].

For the cutting pattern selection the following algorithms are commonly used random forest, decision tree, structured kernel ridge regression, and k-nearest neighbors. As an input features the following characteristics of the log can be used: volume, length, diameter, curvature, and shrinking [9].

Other sawmill problems are also solved using machine learning, for example: modelling the internal distribution of knots [3], timber tracing [13], wood species identification [14], detection of mechanical damage on the surface of a wooden board [15].

2.3 Point cloud processing for sawmilling

In the previous study, a point cloud filtering method was proposed. The method was applied to the data that was provided for this work. An individual log from these data consists of a set of points described by three coordinates x , y , z . A subset of points with the same z value forms a layer resembling a circle. The layers are parallel to each other forming a surface similar to a cylinder. A layer usually contains points belonging to both classes, but log points are supposed to be the majority. Given this, the DBSCAN clustering algorithm is applied to each layer to find the cluster with the biggest number of points. The points of this cluster are considered to belong to the log when the number of points is greater than specific value. If this is not the case, all points belonging to the layer are excluded from the data. The next step is to find points that belong to a log but are considered different clusters by DBSCAN. To do this, a circle is fitted to the previously found cluster using the least squares method. Then, the points located within the specified range from the circle are considered to be belonging to the log. It should be noted that the circle fitting can fail if an error exceeds the specific value. In this case, the entire layer is also excluded from the data [3].

3 DEEP LEARNING

3.1 Background

Deep learning is a subset of machine learning methods based on the neural networks which represents the data as a nested hierarchy of concepts. This hierarchy allows to learn complicated concepts by building them from simpler abstract representations [16]. Neural networks originated in 1943 as an attempt to find mathematical model for information processing in biological systems [17].

Nowadays, deep learning has become a powerful framework for supervised learning [16]. It is the most frequently used method for machine translation, speech synthesis and recognition, visual object recognition, and image synthesis [18]. The main contributor to the popularity of deep learning has been the significant performance gains made possible by the use of Graphics Processing Unit (GPU) [16]. Most problems that consist of mapping an input vector to an output vector can be solved using deep learning, given sufficiently big models and datasets [16].

3.2 Common components of neural networks

A neuron is a basic unit of neural network that calculates the weighted sum of the inputs, adds constant (bias) to the sum and then applies a nonlinear function (activation function) to produce its output. The nonlinearity allows a sufficiently large neural network to represent arbitrary continuous functions [18]. This is why neural networks also called universal approximators [19].

The neurons (nodes) can be arranged in layers of arbitrary size. The network can be composed of different types of layers with different types of activation functions. The weights represent the strength of connections between layers [20]. An activation function can be for example Rectified Linear Unit (ReLU), tanh, sigmoid [18]. The structure of an elementary neural network is demonstrated in Fig. 3.

The input layer receives the data, which is then used by the neural network. The length of this layer depends on the number of features that the data contains [20]. The last layer of the neural network (output layer) produces the end result. Depending on the type of task solved by the neural network, the last layer can be represented by one neuron when

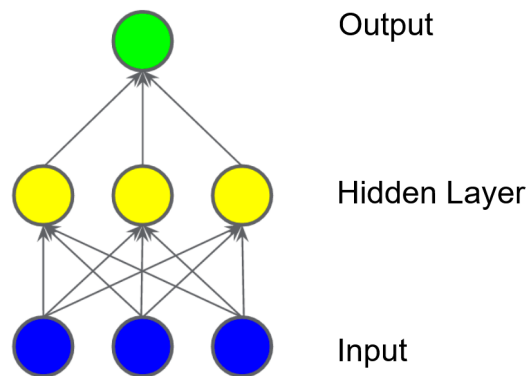


Figure 3. The structure of an elementary neural network [20].

the problem is regression, one or more neurons with a Softmax activation function when the problem is classification [16]. The hidden layers are all layers between the input and output layers [20].

Given the input data the neural network makes a prediction. This is done through the process called forward propagation (or feed-forward operation). During the propagation the data is traversed through the network towards the output layer from the input layer [16].

3.3 Neural network training

Neural network training can be described as an optimization problem, which consists in selecting the optimal weights and biases of neurons throughout the network. The current approach to training neural networks is based on an iterative optimization algorithm called gradient descent [16].

In order to estimate how much the network's predicted values differ from the expected values a loss function (also known as a cost function) is used. An example of loss functions for a regression problem is Mean Squared Error (MSE) and cross entropy for classification [16].

The training consist of the following steps [18]:

- Calculation of the gradient of the loss function with respect to the weights.
- Changing the weights along the gradient direction to minimize loss function.

The gradient is computed by passing information back through the network starting from

the output layer. This process is called backpropagation [18].

3.4 Multilayer perceptron

Multilayer Perceptron (MLP), also known as a fully connected feed-forward neural network, is an essential and most basic type of neural network [16]. In a fully connected network, each node in a layer is connected to each node in the subsequent layer [16]. Connections between MLP nodes form a directed acyclic graph between the input and output nodes [18]. In practice, multilayer perceptron is a mathematical function that maps the input values to the output value [16].

Forward propagation for a single neuron of MLP can be represented by the mathematical formula as follows

$$f \left(\sum_{i=1}^M w_i x_i + b \right) \quad (1)$$

where x_i is an activation value which was received from the previous layer or the input, w_i is a strength of the connection x_i , b is a bias, M is a length of the previous layer and f is an activation function [17].

3.5 Convolutional neural network

Convolutional Neural Network (CNN) is a special type of neural network that has become widespread in the field of image processing. CNN is usually applied to the grid-like data such as 2-D matrices. Compared to MLP, neurons in CNN are only connected to a small area of the previous layer. This makes CNN a more computationally efficient alternative to MLP in the image processing domain. Another notable aspect is that the CNN automatically extracts features from the data during the training phase. In contrast, MLP works with handcrafted features [16].

Convolutional neural networks in at least one of their layers use the convolution operation instead of the general matrix multiplication. During convolution, the kernel (filter) moves through the input matrix and performs the dot product. The dot product can be represented

by the mathematical formula as follows

$$\sum_{i=1}^M \sum_{j=1}^N a_{ij} b_{ij} \quad (2)$$

where a is a matrix that represents the sub-region of input data, b is a matrix that represents kernel, i is an index of row in the matrix, j is an index of column in the matrix, M is the number of rows in the matrix, N is the number of columns in the matrix. The dimensions of the kernel matrix and the tile matrix are equal. Convolutional kernels are typically initialised with random numbers and then the network trains the ideal values [20].

The result of convolution is the matrix of dot products. An example of performed convolution is shown in Fig. 4. The size of the input feature map is 5x5, the size of the convolution kernel is 3x3, and the size of the resulting feature map is 3x3.

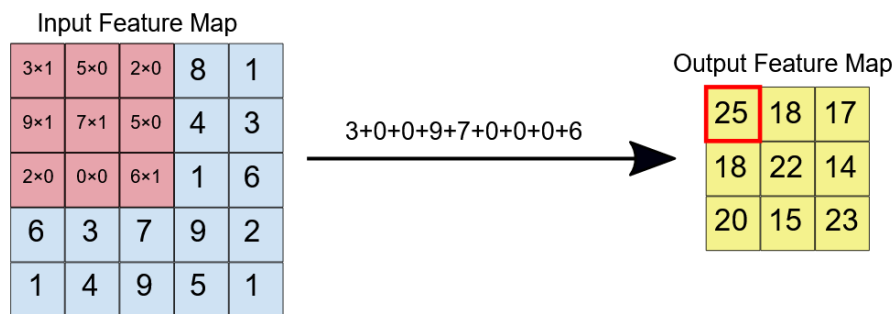


Figure 4. Convolution operation [20].

After convolution, an activation function is applied to the extracted features. Next, a pooling operation is usually applied to the convolved features. The pooling is needed to reduce the size of the feature map and improve invariance to translation of the CNN. There are different types of pooling, but the most widely used is max pooling. As in the case of convolution, the pooling kernel moves across the feature map and extract tiles. In every extracted tile, the biggest value is selected for a new feature map [20]. An example of applying the pooling operation is demonstrated in Fig. 5.

Similar to MLP, the last layers of CNN are usually fully connected. Their job is to produce the output value from the features extracted in the previous layers. However, there is another approach that is often used in the segmentation task, it is called Fully Convolutional Network (FCN). The last layers of which are convolutional with 1 by 1 convolution kernels. It allows to give a prediction (e.g. class label) to each pixel of the image [20].

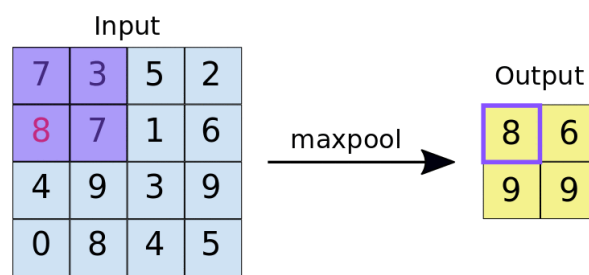


Figure 5. Max pooling operation [20].

4 POINT CLOUD SEGMENTATION

4.1 Background

3-D point cloud is a type of data structure that stores Three Dimensional (3-D) geometric data. The data is usually not ordered and spatially uncorrelated. Point cloud semantic segmentation is the process of assigning a class to each point from the point cloud. The existing approaches can be classified as follows [21]:

- Intermediate representation based:
 - Projection-based: project 3-D point cloud into Two Dimensional (2-D) intermediate representation such as spherical and multi-view representations.
 - Discretization-based: transform a point cloud in a dense/sparse discrete intermediate representations such as sparse permutohedral lattices and volumetric.
- Point-based: work on raw 3-D point cloud. Existing approaches: RNN-based methods, point convolution methods, pointwise MLP, graph-based methods.
- Hybrid: use the combination of aforementioned approaches.

Most algorithms are not suitable for large point clouds. They usually work with point clouds no larger than few thousand points [21]. The classification of intermediate representation based approaches is shown in Fig. 6. The image shows a general idea of how those approaches work. Multi-view representation based algorithms capture a 3-D scene from multiple 2-D perspectives. Spherical representation based algorithms apply spherical projection to different points of a 3-D scene. The resulting images are also 2-D. Dense discretization based algorithms transform a 3-D scene into a 3-D grid cell where the value of each cell is equal to the point density of the corresponding area of the original representation. Sparse discretization based algorithms embed a 3-D scene into more memory and computationally efficient sparse lattice.

The existing point-based approaches can be classified as shown in Fig. 7. The point-wise MLP methods on this image represented by the set abstraction layers of PointNet++. The layers are used to process point features hierarchically [22]. Point convolution methods are represented by the continuous convolution operator which is a part of Deep Parametric Continuous CNN. The method focuses on adapting discrete CNN to work with the point clouds. In the image g is a MLP used as a kernel function, x is a point selected for

convolution, and y_k is a k input points [23]. RNN-based methods are represented by bi-directional RNN which is one of the layers of 3P-RNN method [24]. Graph-based methods are represented by superpoint graph which is a part of SPG method. The superpoint graph takes as input superpoints which are small basic geometric structures [25].

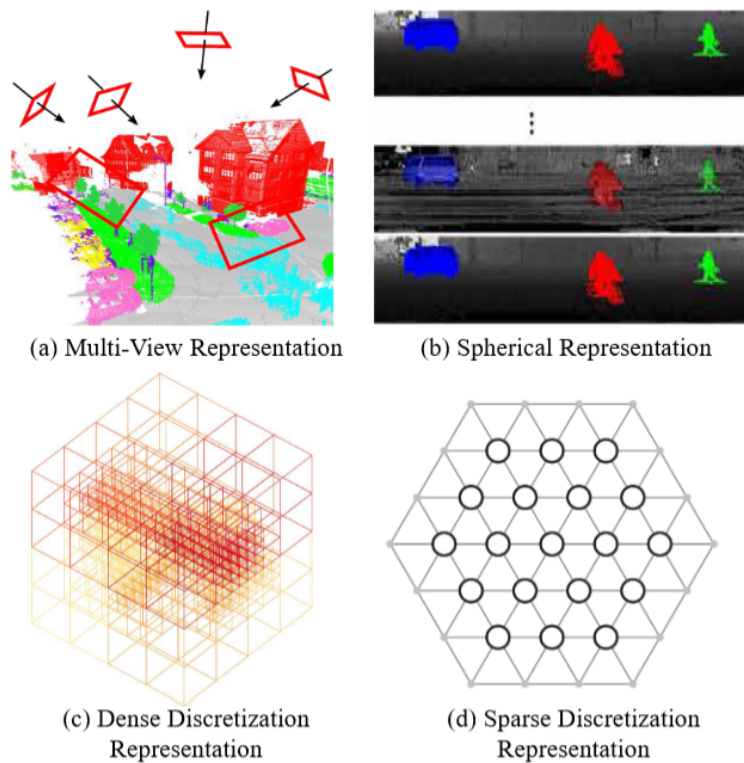


Figure 6. Categorization of approaches that use an intermediate representation [21].

4.2 Projection-based approaches

The first step of any projection-based method is to transform a 3D point cloud to an 2D intermediate regular representation. This allows to apply the common 2-D image segmentation methods such as various Convolutional Neural Network architectures. In comparison with point-based methods projection-based approaches usually shows better inference speed. One major drawback of these methods is the loss of information. This is usually caused by 3D-2D projection [21].

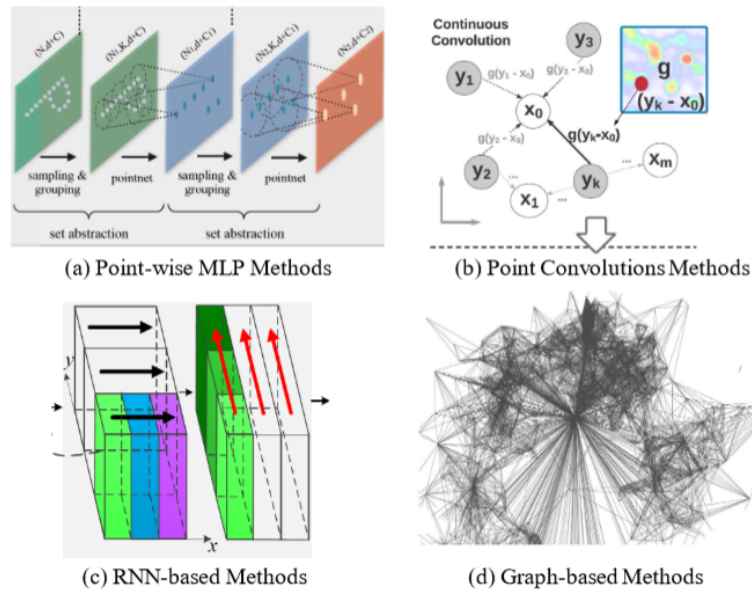


Figure 7. Categorization of the point-based methods [21].

4.2.1 Multi-view representation

The methods of this category use multiple camera positions to predict pixel-wise score. The final semantic label is produced by fusing the scores. The performance of these methods is affected by viewpoint selection and occlusions. Multi-view representation methods have not fully exploited the underlying structural and geometric information. The projection inevitably introduces information loss [21].

One example method from this category is TangentConv. This approach applies virtual camera to each point of a point cloud. In practice each camera is a tangent plane located in the scene so that points in the defined radius can be projected on this plane. The resulting 2-D image is processed using convolution layers. The network type is FCN encoder-decoder [26]. Also, there is an assumption that input point clouds are sampled from locally Euclidean surfaces. The advantages and disadvantages of this method presented in Table 1.

4.2.2 Spherical representation

The projection step always brings problems such as discretization errors and occlusions. However, in comparison to a single view projection, spherical projection keeps more information. Methods of this category usually show better accuracy and performance [21].

Table 1. Comparison of Multi-view Representation methods.

Name	Pros & Cons
TangentConv [26]	<ul style="list-style-type: none"> +Scalability (can process millions of points) +Applicable to the noisy real-world data +There is an open-source implementation +High accuracy +Efficient in terms of processing speed

As an example, consider RangeNet++. The method is specifically designed to process real-time data from a rotating LiDAR sensor to be used on autonomous vehicles. The first stage of the method is a conversion of a point cloud from a 3D Cartesian coordinate system into a 2D range image. Then the data is transferred to FCN encoder-decoder. After that, the resulting semantic labels and input range image are re-projected back to the 3D Cartesian coordinate system. This is followed by post-processing to eliminate artifacts introduced by the encoder-decoder network and first-stage re-projection [27]. The advantages and disadvantages of RangeNet++ are listed in Table 2.

Table 2. Comparison of Spherical Representation methods.

Name	Pros & Cons
RangeNet++ [27]	<ul style="list-style-type: none"> +Fast inference +Scalability +High accuracy +There is an open-source implementation

4.3 Discretization-based approaches

As projection-based methods, the discretization-based methods first transform the point cloud into an intermediate regular representation. The main disadvantage of discretization-based methods is the cubic increase in computational and memory costs caused by the increase in resolution [21].

4.3.1 Dense discretization representation

The volumetric representation preserves the neighborhood structure of point cloud. Early methods often voxelized the point clouds as dense grids and then applied 3D convolutions. Every point in the voxel is assigned the same semantic label as the voxel. The volumetric representation is naturally sparse, since the number of non-zero values is only a small percentage. It is computationally inefficient to apply dense convolution neural networks to the spatially sparse data [21].

The first step is the voxelization, which usually causes discretization artifacts and information loss. In practice, it is not trivial to choose an appropriate grid resolution. High resolution leads to high memory and computational costs. Low resolution usually results in loss of detail, which subsequently leads to lower accuracy [21].

Methods that utilize 3D CNN or FCN usually show a good scalability. These methods can be trained and tested on point clouds with different spatial sizes [21]. Some prominent methods, their advantages and disadvantages are presented in Table 3.

Table 3. Comparison of Dense Discretization Representation methods.

Name	Pros & Cons	Notes
FCPN [28]	<ul style="list-style-type: none"> +Operates on raw sensor data, no pre-processing/encoding necessary +Can process data with a large number of points +It shows good scalability during inference +It can be trained on smaller training dataset, then be scaled up during inference to process spaces multiple times larger than it was trained on in a single shot +There is an open-source implementation 	-
VV-NET [29]	<ul style="list-style-type: none"> -No open-source implementation +Effective with noisy data, robust for learning 	Uses variational autoencoder and radial basis function

4.3.2 Sparse discretization representation

In comparison with the volumetric representation methods of this category greatly reduce computational and memory overhead by limiting the output of convolution to be only related to occupied voxels [21]. Some prominent methods, their advantages and disadvantages are presented in Table 4.

Table 4. Comparison of Sparse Discretization Representation methods.

Name	Pros & Cons
SPLATNet [30]	+The method outperformed state-of-the-art algorithms in 2018 +There is an open-source implementation
LatticeNet [31]	+Less quantization issues +Improved accuracy +It can process large point clouds +There is an open-source implementation

4.4 Point-based approaches

Point representation does not have explicit neighboring information. Most modern point-based algorithms utilize expensive neighbor searching methods such as ball query and k-nearest neighbors. This limits the efficiency of these algorithms, although there are some promising approaches [21].

4.4.1 Shared multilayer perceptron methods

Pointwise features extracted using shared MLP cannot capture the mutual interactions between points and local geometry. However, methods of this category usually show high efficiency [21].

In order to improve accuracy several techniques have been introduced, including methods based on attention-based aggregation, local-global feature concatenation, and neighboring feature pooling [21].

In order to capture local geometric patterns, methods that utilize neighboring feature pooling learn a feature for each point by aggregating the information from local neighboring points. Some approaches have achieved high efficiency in terms of memory and computation [21]. To further improve segmentation accuracy local-global concatenation methods utilise permutation-invariant neural network architectures to incorporate local structures and global context from point clouds [21]. Some prominent methods, their advantages and disadvantages are presented in Table 5.

4.4.2 Point convolution methods

These methods utilize computationally effective convolution operators [21]. 3D CNNs dominated early stage research of deep learning on 3D vision, where the point clouds were represented into 2D multi-view images or 3D voxels [33]. Some prominent methods, their advantages and disadvantages are presented in Table 6.

4.4.3 Recurrent neural network methods

In order to capture inherent context features from point clouds, Recurrent Neural Network (RNN) has also been proposed for 3-D point cloud semantic segmentation. Experimental results with these methods show that incorporation of spatial context is important for the improvement of the segmentation performance. However, some of these algorithms lose rich geometric features and density distribution from the data [21].

One example method from this category is previously mentioned 3P-RNN. The method first divides the input point cloud into overlapping subspaces with the same length and width, aligned by the height of the point cloud. The resulting subspaces have the shape of rectangular cuboids. Then, shared MLP is used to obtain point features for each subspace. To take into account the local context information of each point, the pyramid pooling is applied to each subspace. The resulting features are concatenated with the point features and then fused by a convolutional layer. Next, the subspaces' features are processed along the x-axis by a multi-RNN layer. The resulting features are reassembled and passed to the following multi-RNN layer to process them from the y-direction. Though it wasn't implemented, the processing along the z-axis is possible. In the end, the three feature types are then concatenated and used to predict the label of each point [24].

Table 5. Comparison of Pointwise MLP methods.

Name	Pros & Cons	Notes
PointNet++ [22]	+State-of-the-art in 2017 +Robustness to sampling density variation +There is an open-source implementation	Neighboring feature pooling
RandLA-Net [32]	+Efficient processing of large-scale data +Computation and memory efficient lightweight network (can process 1,000,000 points in a single pass with up to 200X faster than existing approaches) +State of the art in 2019 +There is an open-source implementation	Neighboring feature pooling
Yang, Jiancheng, et al. [33]	-There is no open-source implementation +The method is able to select a representative subset of data +More robust and less sensitive to outliers	Attention-based aggregation
LSANet [34]	+The method better captures the spatial distribution of the point cloud. +Close to the state of the art in 2019 +There is an open-source implementation	Attention-based aggregation
EdgeConv [35]	+The method captures the local information and scene-level global features. +Acts on graphs dynamically computed in each layer of the network. It is differentiable and can be plugged into existing architectures +Shows state-of-the-art performance +There is an open-source implementation (more than 12 implementations)	Local-global concatenation

Table 6. Comparison of Point Convolution methods.

Name	Pros & Cons
Pointwise [36]	-Poor performance in comparison with other methods +There is an open-source implementation
KPConv [37]	+High robustness under varying densities of point clouds +State-of-the-art on several datasets in 2019 +There is an open-source implementation

4.4.4 Graph-based methods

In order to capture the basic shapes and geometric structures of 3-D point clouds, several methods resort to graph networks. Some approaches represented a point cloud as a set of interconnected simple shapes and superpoints, and used an attributed directed graph to capture the structure and context information. Several methods have shown that they can operate under weak supervision [21]. Some prominent methods, their advantages and disadvantages are presented in Table 7.

Table 7. Comparison of Graph-based methods.

Name	Pros & Cons
MPRM [38]	+Weakly supervised approach +There is an open-source implementation
Xu, Xun, and Gim Hee Lee. [39]	+The network can be trained with only partially labeled points (e.g. 10%) +There is an open-source implementation

4.5 Hybrid methods

The methods of this category use combination of previously described approaches. The main advantage of hybrid methods that they leverage all available information from the data [21]. As an example, consider UPB method. The network type is encoder-decoder with skip connections. First, the method splits the input point cloud into subspaces of

equal volume. Then, it consecutively processes them by combining features from input point cloud, point cloud in the current subspace, and 2D images captured from different positions in the subspace. Features from 2D images extracted using deep convolutional neural network. To extract local geometric features from the subspace and global context from the input point cloud, the algorithm directly applies point-based networks. The method is able to work with 3D meshes and color information [40]. More information about the advantages of hybrid methods is presented in Table 8.

Table 8. Comparison of Hybrid methods.

Name	Pros & Cons	Notes
UPB [40]	+No voxelization +In terms of accuracy the method outperforms several state-of-the-art approaches in 2019 +There is an open-source implementation	Multi-view + point based network
MVPNet [41]	+No voxelization +Significantly outperforms previous approaches in 2019 +There is an open-source implementation	Multi-view + point based network

5 3D POINT CLOUD SEMANTIC SEGMENTATION

5.1 Justification of choice

The point-based RandLA-Net method [32] was chosen for the following reasons:

- Many point cloud segmentation algorithms can only work with a very small amount of data (about 4000 points). In contrast, RandLA-Net can work with large-scale point clouds (millions of points). Which is an important feature because, on average, the point cloud in this research consists of 500,000 points.
- Unlike some of the previous methods (e.g. PointNet), RandLA-Net does not split the input data into blocks to separately process each of them. Instead, it operates on the entire data and considers the whole geometry.
- The network works on the raw data and does not utilize pre-processing or post-processing methods.
- The network is computationally and memory efficient.
- The network makes accurate predictions even for data with complex geometric structures.

5.2 Architecture

The main technique that allows RandLA-Net to process large point clouds is downsampling applied to each layer of the network. Random sampling is used as a downsampling method, mainly because of its computational efficiency. The only problem is that random sampling can accidentally discard important features.

To solve this problem Local Feature Aggregation module is proposed (see Fig. 8). This module progressively increases the receptive field for each 3D point. The module is applied in parallel to each point. It consists of 3 neural units: Local Spatial Encoding, Attentive Pooling, and Dilated Residual Block. The module utilizes only feedforward neural networks.

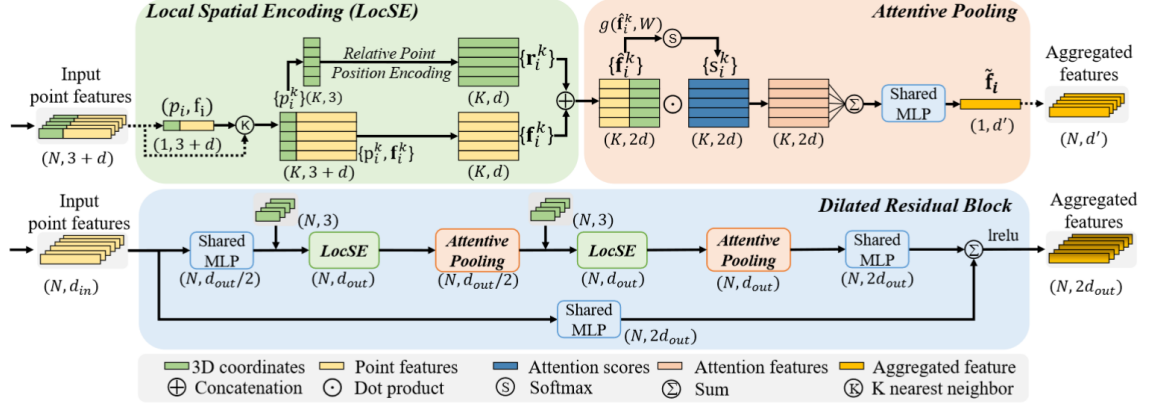


Figure 8. Local Feature Aggregation module [32].

5.2.1 Local Spatial Encoding

Local Spatial Encoding unit (see Fig. 8) allows to learn complex geometric patterns in the data. The module performs the following sequence of actions:

- It finds neighboring points for each point in the input data. To do this, the module uses the K-nearest neighbors algorithm with Euclidean distance as the distance metric. The found points can be represented by the mathematical formula as follows

$$\{p_i^1, \dots, p_i^k, \dots, p_i^K\} \quad (3)$$

where p is the coordinates of one of the nearest points to the i^{th} center point, i is the index of a center point. The value of index ranges from 1 to the total number of points in the input data. K is a predefined number of nearest points, and k is the index of a neighbouring point.

- Then the relative point positions of all neighboring points are being encoded so that the corresponding point features are always aware of their relative spatial locations. This step can be represented by the mathematical formula as follows

$$r_i^k = MLP(p_i \oplus p_i^k \oplus (p_i - p_i^k) \oplus \|p_i - p_i^k\|) \quad (4)$$

where p_i is the coordinates of a center point, p_i^k is the coordinates of a neighboring point, $\| \cdot \|$ is the Euclidean distance between center and neighbouring points, \oplus is the concatenation operator, r_i^k is the encoded relative point positions.

- Then the encoded relative point positions are concatenated with its corre-

sponding point features. This step can be represented by the mathematical formula as follows

$$\hat{f}_i^k = r_i^k \oplus f_i^k \quad (5)$$

where \hat{f}_i^k is augmented feature vector, f_i^k is point features.

- The final result of this module is the set of augmented feature vectors. The results can be represented by the mathematical formula as follows

$$\hat{F}_i = \{\hat{f}_i^1, \dots, \hat{f}_i^k, \dots, \hat{f}_i^K\} \quad (6)$$

5.2.2 Attentive Pooling

Attentive Pooling module (see Fig. 8) is used to aggregate the neighboring feature set. The module performs the following sequence of actions:

- The module utilizes shared MLP and Softmax function to learn attention score for each augmented feature. This step can be represented by the mathematical formula as follows

$$s_i^k = g(\hat{f}_i^k, W) \quad (7)$$

where W is the weights of the shared MLP, $g()$ is the function that consist of shared MLP and Softmax function, and s is the attention scores.

- This attention score is then used as a soft mask that automatically selects important features. This step can be represented by the mathematical formula as follows

$$\tilde{f}_i = \sum_{k=1}^K (\hat{f}_i^k \cdot s_i^k) \quad (8)$$

where \tilde{f}_i is the feature vector containing the most important features.

5.2.3 Dilated Residual Block

Dilated Residual Block (see Fig. 8) is designed to increase the receptive field (neighborhood) for each point. The module helps to preserve geometric details of input data, which might be otherwise discarded due to the random sampling. The module consists of two

Local Spatial Encoding and Attentive Pooling units, which are stacked with a skip connection. The module makes it possible to obtain information from up to K^2 neighbouring points in two iterations of applying Local Spatial Encoding/Attentive Pooling units. The Fig. 9 shows the example of expanding of the receptive field through the feature propagation.

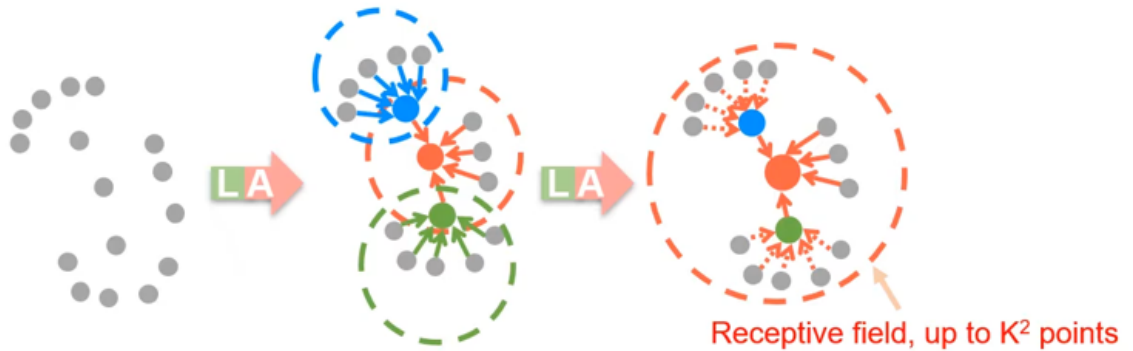


Figure 9. Dilation of the receptive field [32].

Although the number of Local Spatial Encoding and Attentive Pooling units can be chosen arbitrarily, the number 2 was chosen by the authors as optimal due to the balance between computational performance, accuracy, and over-fitting.

5.2.4 RandLA-Net

RandLA-Net consists of several units. It starts with a fully connected layer. Which is followed by 4 encoding and decoding layers, followed by the last three fully connected layers. This type of neural network architecture called encoder-decoder architecture with the skip connections. The Fig. 10 shows the detailed network architecture.

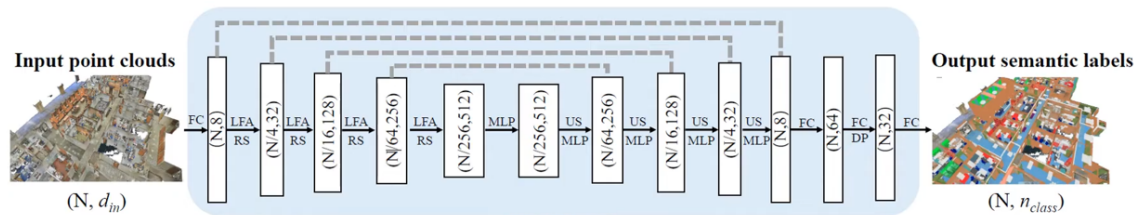


Figure 10. Architecture of RandLA-Net [32].

The initial fully connected layer is used for feature extraction. It takes as input N by d matrix, where N is the number of points in dataset, d is a feature dimension which includes 3D coordinates and optionally, RGB color information.

The encoders are utilized to decrease the size of point clouds while increasing the point feature dimension. They consist of stacked Local Feature Aggregation and Random Sampling layers. As a result, the point cloud is reduced by a factor of 256, and the point feature dimension is increased by a factor of 512. The decoders first upsample each point, then add the features from the encoding layers, and finally apply a fully connected layer.

The final three fully connected layers are used to predict semantic label of each point. A dropout layer is also used to improve accuracy. The last fully connected layer outputs N by n_{class} matrix, where N is the number of points in dataset, n_{class} is predicted class label (the value of which lies in the range from 0 to the number of classes minus one) [32].

6 EXPERIMENTS

6.1 Data

There were used 3 datasets for the experiments. The points were captured using laser scanners on the Scots pine softwood logs. The age of the wood is different. The scanning of logs for Datasets 1 and 2 was performed on the same sawmill. Dataset 3 was gathered at the separate sawmill. The datasets contain similar data (logs, metal railings, noise). The examples of point clouds from each dataset are shown in Fig. 11.

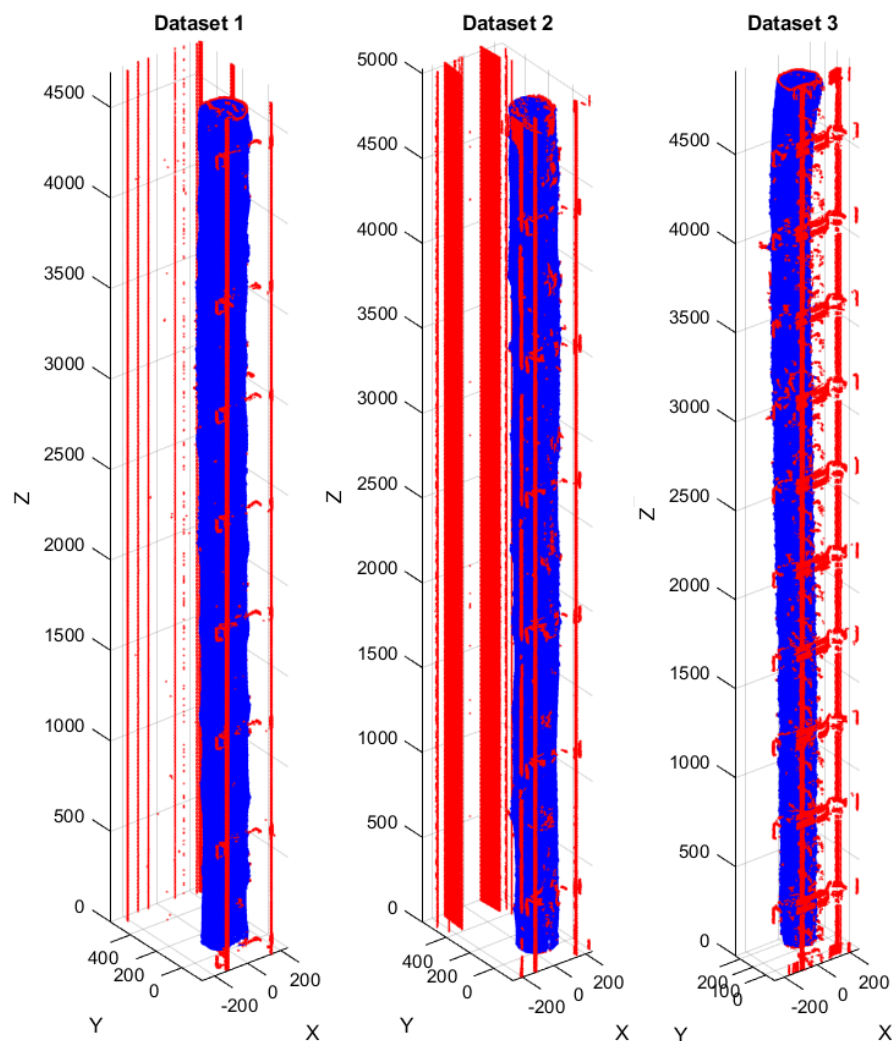


Figure 11. Visualization of datasets.

The datasets are unbalanced. There are considerably more points that belong to logs than points that belong to noise. The logs from Dataset 1 and Dataset 2 were scanned multiple

times. The number of 3D points per file varies from 300,000 to 700,000. More detailed information on the quantitative characteristics of these datasets is presented in Table 9.

Table 9. Characteristics of datasets.

Name	Number of files	Number of logs	Scans per log	Ratio (log / noise)	Points per file
Dataset 1	200	50	4	90% / 10%	600,000
Dataset 2	100	50	2	70% / 30%	700,000
Dataset 3	100	100	1	87% / 13%	350,000

The labels for these datasets were assigned using the DBSCAN clustering algorithm, so there are some points that were assigned incorrectly.

6.2 Description of experiments

The following experiments were carried out using RandLA-Net model:

- Experiment 1. A single model was trained on a randomly selected 80% of the data from all datasets. Then the model was tested on the rest 20% of the data from all datasets. Also, a visual explanation of the experiment is shown in Fig. 12.

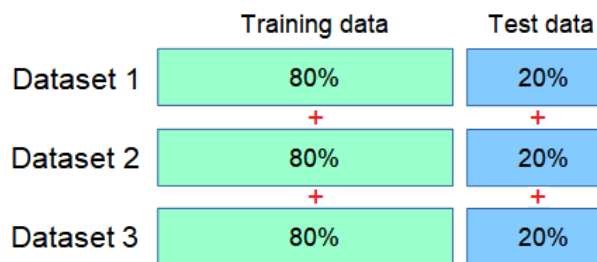


Figure 12. Experiment 1.

- Experiment 2. Models were trained in a 3-fold cross-validation manner. For example, one model was trained on Datasets 1 and 2, then it was tested on the Dataset 3. After that next model was trained on Datasets 1 and 3, then it was tested on the Dataset 2, and so on. This experiment tests the ability of the model to generalize depending on the training data. A visual explanation of the experiment is shown in Fig. 13.

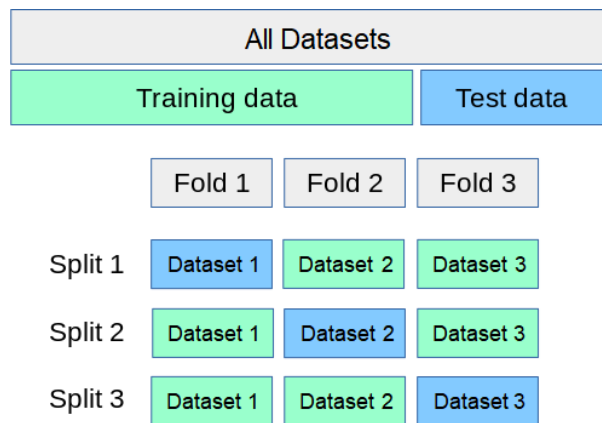


Figure 13. Experiment 2.

- Experiment 3. This experiment is similar to the Experiment 2, but now only one dataset is used for training, and the other two are used for testing. A visual explanation of the experiment is shown in Fig. 14.

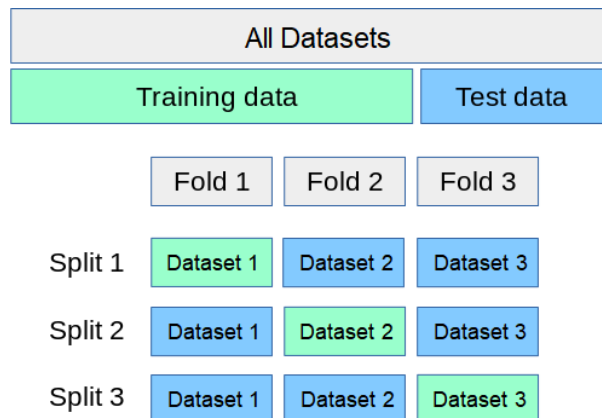


Figure 14. Experiment 3.

The official RandLA-Net implementation from [42] was used to conduct experiments. Python scripts `helper_tool.py`, `main_S3DIS.py`, `6_fold_cv.py`, `data_prepare_s3dis.py` and text files `anno_paths.txt`, `class_names.txt` have been adapted to work with current datasets.

The following training hyperparameters were changed from their default values. The number of training epochs was set to 5. The number of validation steps was set to 100. The size of the validation batch was set 6. The number of classes was set to 2.

The hardware and software components that were used for conducting experiments are presented in Table 10.

Table 10. The hardware and software components.

CPU	Intel Xeon E5-2680 @ 2.70GHz
GPU	NVIDIA GeForce GTX 1080 Ti
RAM	256GiB
OS	Ubuntu 20.04.4 LTS
Python	3.6.13
Tensorflow	1.11.0

6.3 Evaluation criteria

The confusion matrix was used for the evaluation. Logs are considered a positive class and noise a negative class. The following terms are used in further accuracy measures.

- True Positive (TP) - positive classes that were predicted as positive classes
- False Positive (FP) - negative classes that were predicted as positive classes
- True Negative (TN) - negative classes that were predicted as negative classes
- False Negative (FN) - positive classes that were predicted as negative classes

In order to evaluate results of semantic segmentation the following metrics were used:

- Precision. It shows the ratio of correctly predicted positive classes among the classes that were predicted to be positive. This measure can be represented by the mathematical formula as follows:

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

- Recall. It shows the ratio of correctly predicted positive classes among all positive classes. This measure can be represented by the mathematical formula as follows:

$$Recall = \frac{TP}{TP + FN} \quad (10)$$

- Accuracy. It shows the ratio of correctly predicted classes among all predictions. It can be a misleading metric for imbalanced data sets. Which is true for this research. This measure can be represented by the mathematical formula as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

- F1-score. It combines precision and recall metrics. This metric is better suited for unbalanced datasets. This measure can be represented by the mathematical formula as follows:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (12)$$

The values of all these metrics are in the range from 0 to 1 (1 is the best case, 0 is the worst).

6.4 Results

In the first experiment, 80% of the data from each dataset was used for training, and the remaining 20% was used for testing. The results of the experiment are presented in Table 11. The calculated confusion matrix is shown in Fig. 15. To visually evaluate the accuracy of the trained model see Fig. 16. The image contains visualized ground truths and predictions for all three datasets.

Table 11. Results of Experiment 1.

Model	Accuracy	Precision	Recall	F1-score
RandLA-Net	0.9938	0.9718	0.9915	0.9816

Confusion matrix

True Class	1	7319006	212350	97.2%	2.8%
	2	62454	36942608	99.8%	0.2%
		99.2%	99.4%		
		0.8%	0.6%		
		1	2		
		Predicted Class			

Figure 15. Confusion matrix: (1) Class Noise; (2) Class Logs.

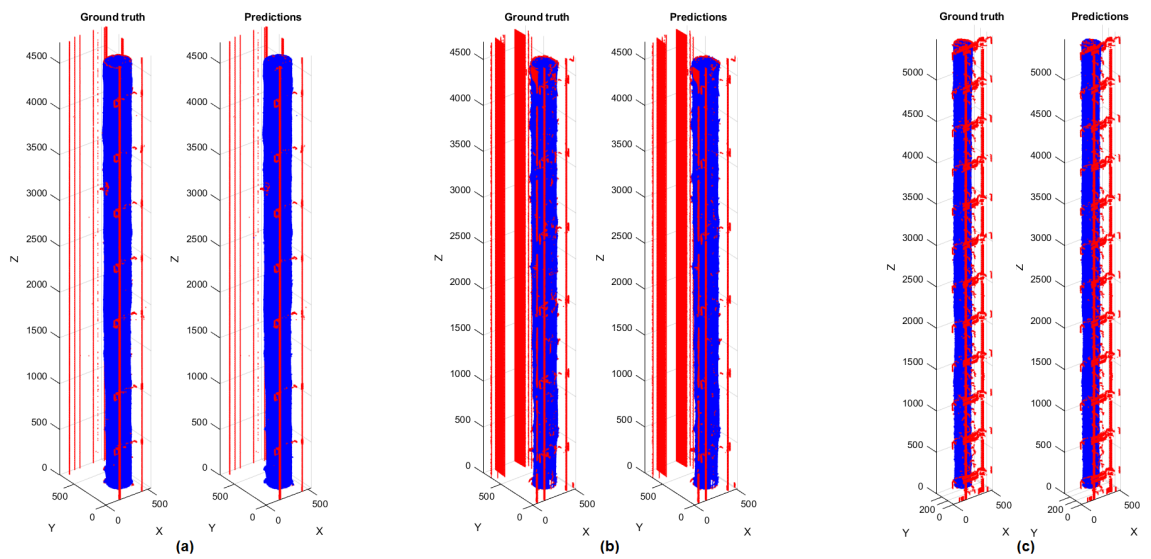


Figure 16. Examples of ground truth and prediction: (a) Dataset 1; (b) Dataset 2; (c) Dataset 3.

In the second experiment, three models were used. Each model was trained on two datasets and tested on the remaining one. The results of the experiment are presented in Table 12. The calculated confusion matrices are shown in Fig. 17. To visually evaluate the accuracy of the trained models see Fig. 18. The image contains visualized ground truths and predictions for all three models.

Table 12. Results of Experiment 2.

Model	Training datasets	Testing dataset	Accuracy	Precision	Recall	F1-score
RandLA-Net	1,2	3	0.9784	0.8770	0.9443	0.9094
	1,3	2	0.8161	0.5526	0.7974	0.6528
	2,3	1	0.9962	0.9698	0.9915	0.9805

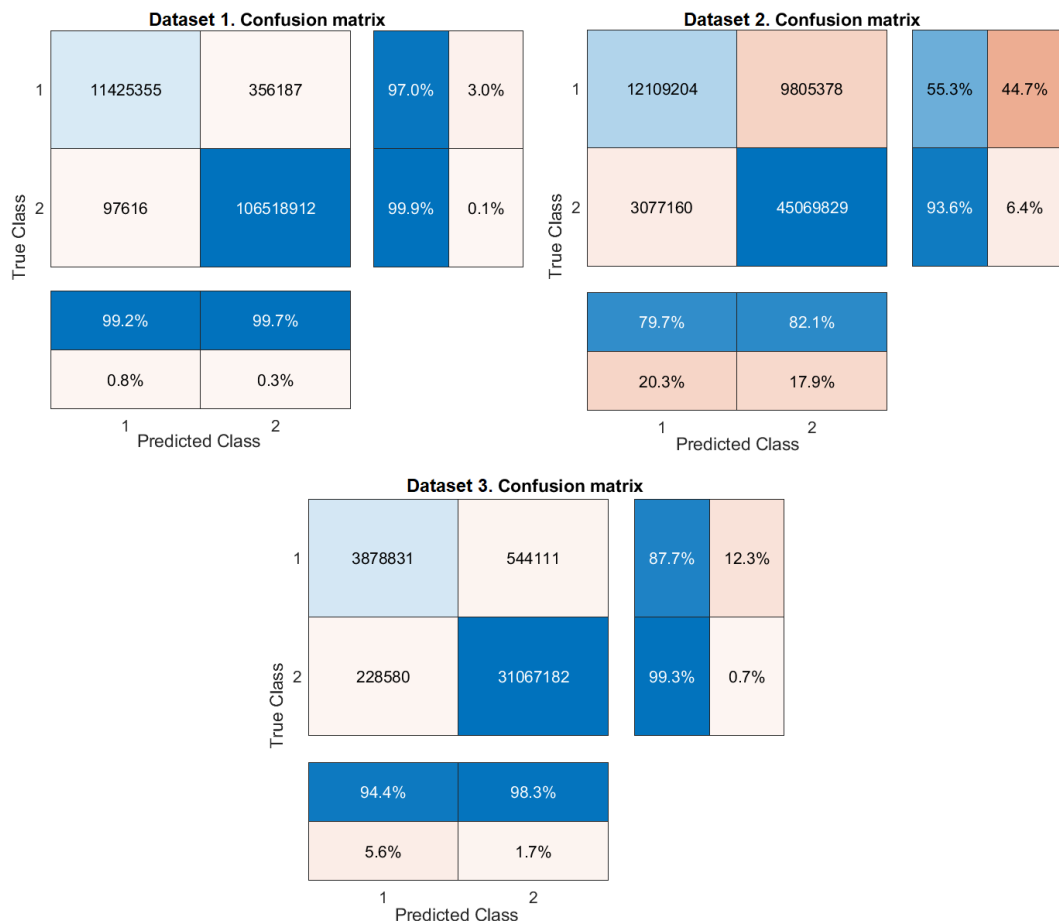


Figure 17. Confusion matrices: (1) Class Noise; (2) Class Logs.

The third experiment is similar to Experiment 2. Three models were used. Each model was trained using only one dataset and the remaining two datasets were used for test-

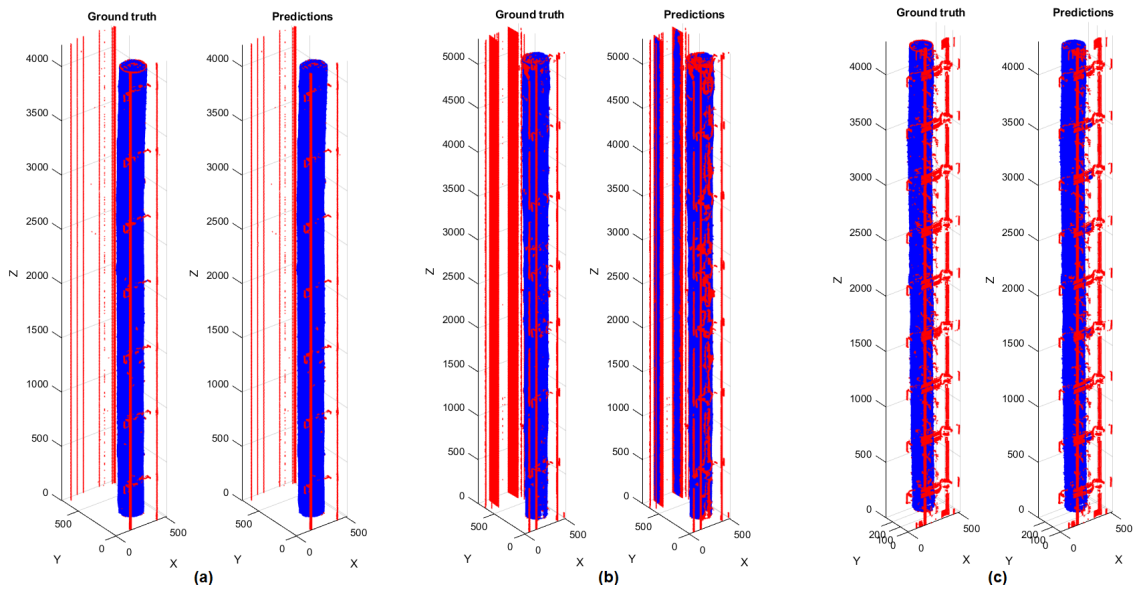


Figure 18. Examples of ground truth and prediction: (a) Dataset 1; (b) Dataset 2; (c) Dataset 3.

ing. The results of the experiment are presented in Table 13. The calculated confusion matrices for the model that was trained only using Dataset 1 are shown in Fig. 19. The calculated confusion matrices for the model that was trained only using Dataset 2 are shown in Fig. 20. The calculated confusion matrices for the model that was trained only using Dataset 3 are shown in Fig. 21. To visually evaluate the accuracy of the model trained using only Dataset 1, see Fig. 22. The image contains visualized ground truth and predictions made on Dataset 2 (two logs on the left) and Dataset 3 (two logs on the right). To visually evaluate the accuracy of the model trained using only Dataset 2, see Fig. 23. The image contains visualized ground truths and predictions made on Dataset 1 (two logs on the left) and Dataset 3 (two logs on the right). To visually evaluate the accuracy of the model trained using only Dataset 3, see Fig. 24. The image contains visualized ground truth and predictions made on Dataset 1 (two logs on the left) and Dataset 2 (two logs on the right).

Table 13. Results of Experiment 3.

Model	Training dataset	Testing dataset	Accuracy	Precision	Recall	F1-score
RandLA-Net	1	2	0.8553	0.5590	0.9629	0.7074
		3	0.9655	0.9166	0.8244	0.8680
RandLA-Net	2	1	0.9964	0.9727	0.991	0.9817
		3	0.9692	0.7846	0.9594	0.8632
RandLA-Net	3	1	0.9948	0.9689	0.9788	0.9738
		2	0.7819	0.3294	0.9250	0.4858

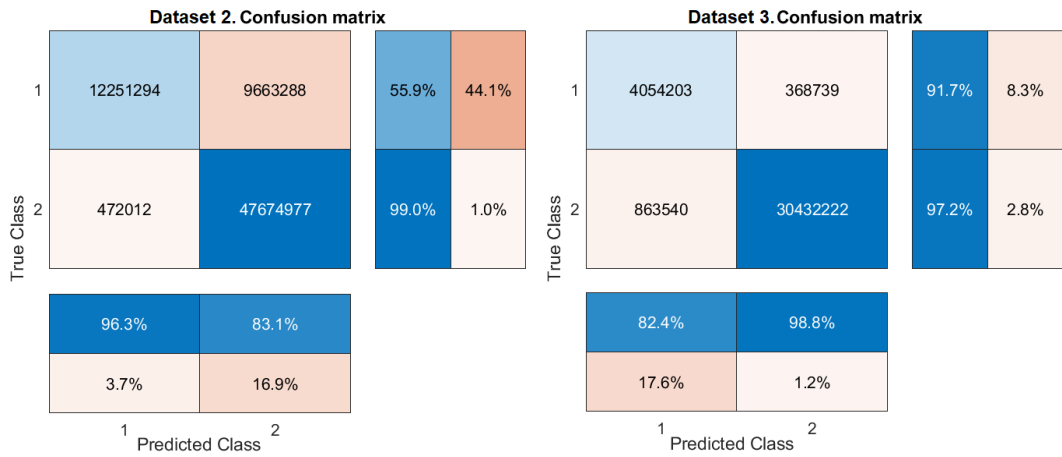


Figure 19. Confusion matrices for the model trained only on Dataset 1: (1) Class Noise; (2) Class Logs.

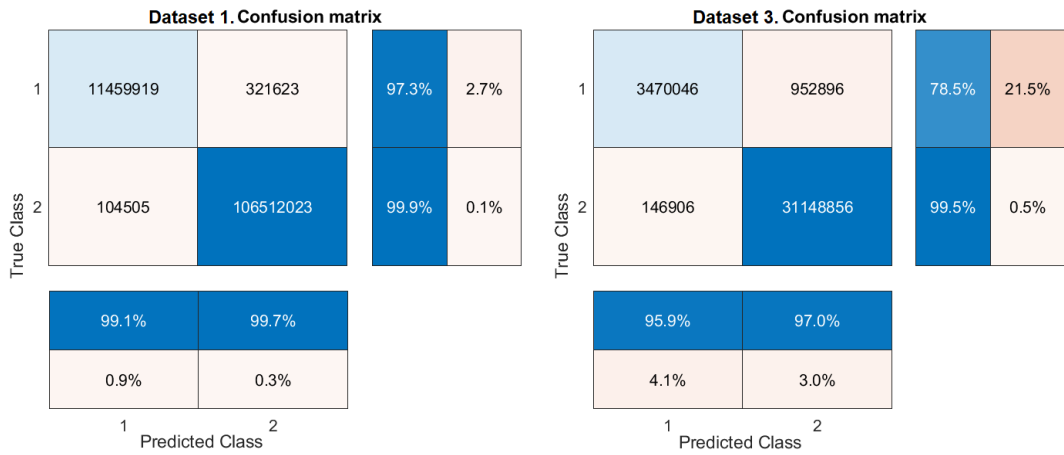


Figure 20. Confusion matrices for the model trained on only Dataset 2: (1) Class Noise; (2) Class Logs.

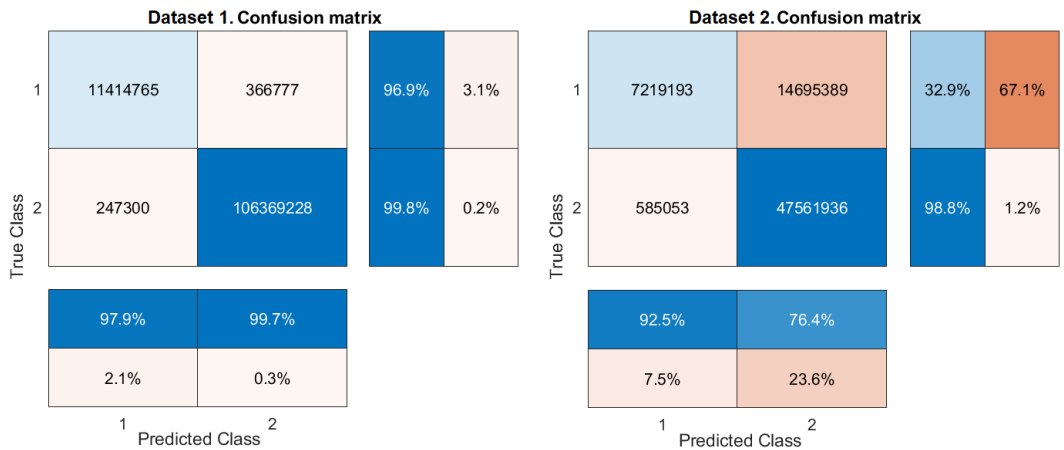


Figure 21. Confusion matrices for the model trained only on Dataset 3: (1) Class Noise; (2) Class Logs.

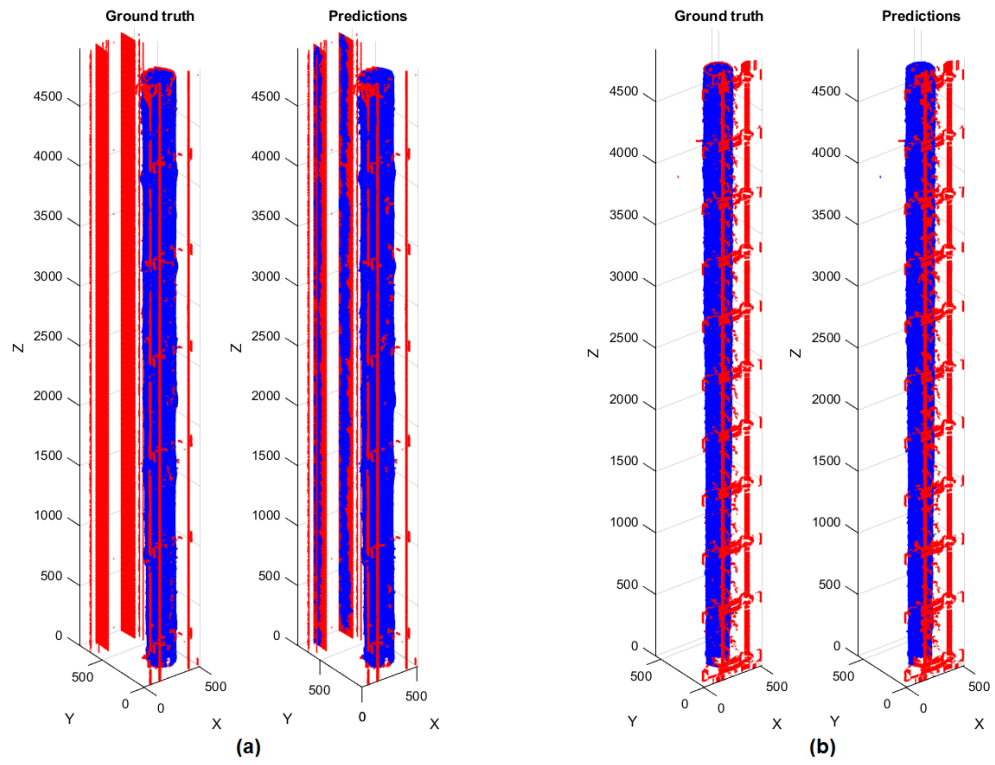


Figure 22. Examples of ground truth and prediction of model trained on Dataset 1: (a) Dataset 2; (b) Dataset 3.

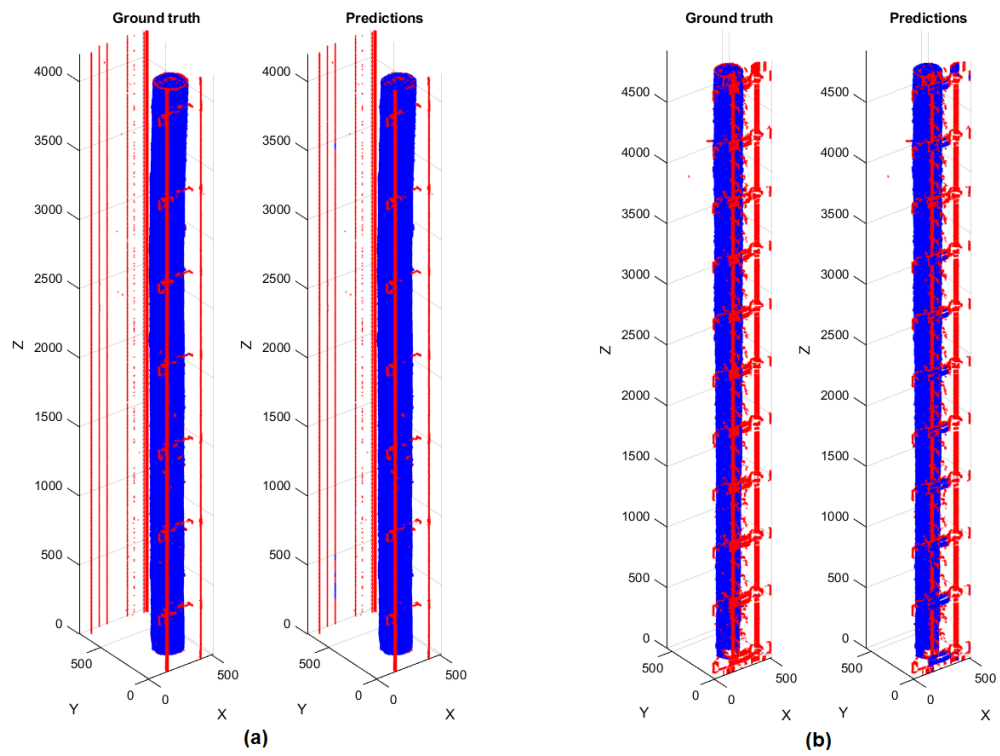


Figure 23. Examples of ground truth and prediction of model trained on Dataset 2: (a) Dataset 1; (b) Dataset 3.

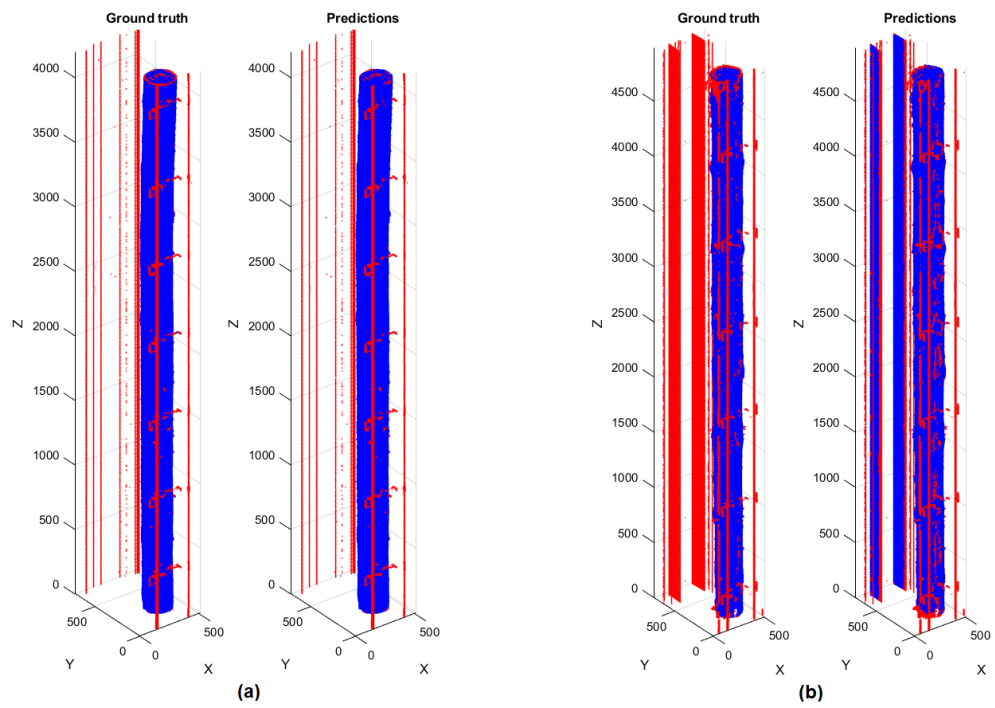


Figure 24. Examples of ground truth and prediction of model trained on Dataset 3: (a) Dataset 1; (b) Dataset 2.

7 DISCUSSION

7.1 Current study

The work is focused on improving the existing solution [3] to the problem of noise filtering in point clouds of wooden logs. After reviewing existing deep learning methods that are suitable for this problem, RandLA-Net was chosen. Three experiments were proposed to evaluate the suitability of this method for solving the previously stated problem.

Experiment 1 was designed to test the basic performance of the method on the previously unseen but relatively similar data. The results show that 2.8% of noise points and 0.02% of log points were misclassified (see Fig. 15). Visual inspection also did not reveal any significant errors. Accuracy and F1-score close to 1.

Experiment 2 was designed to test the ability of the network to generalize. The model is not able to distinguish noise and log for Dataset 2 when it is trained on Dataset 1 and Dataset 3 (see Fig. 17). The results of segmentation on Dataset 1 is comparable to the results achieved in Experiment 1. For Dataset 3 the accuracy and F1-score dropped.

Experiment 3 was designed to further test the ability of model to generalize. As in the previous experiment the network is not able to accurately predict labels of Dataset 2 based on the training on Dataset 1 or Dataset 3 (see Fig. 13). The accuracy and F1-score are lower for Dataset 2 when the network trained on Dataset 3 than when it is trained on Dataset 1. High F1-score was achieved for Dataset 1 regardless of the training dataset.

Summarizing the results of all experiments, it can be concluded that for RandLA-Net to show high prediction accuracy, the training should include samples from Dataset 2. Also, compared to misclassified log points, the fraction of correctly classified noise points is lower in each experiment. This may be a result of an imbalanced dataset.

As described earlier, the data was not labeled manually, since each log consists of hundreds of thousands of points, this would be a laborious process. Therefore incorrect ground truth labels might also have a negative impact on accuracy.

In addition, during those experiments RandLA-Net in practice demonstrated fast inference speed on a single GPU (about a second per log). In the context of a possible industrial application of the method, this can be considered an advantage. The fast training of the model compensates for the limitations of data generalization.

7.2 Future work

Future research may utilize semi-supervised deep learning methods to address the problems associated with ground truth labeling. As the existing approaches [38, 39] require a small subset of data to be labeled for training and show comparable accuracy. In this case, to avoid manual labeling, the previously described DBSCAN-based approach can be used with more stringent thresholds. This will produce less labeled but more accurate training data.

The results of RandLA-Net can also be improved. Greater amount of training data will likely improve accuracy. Handling class imbalance problem by augmentation of noise can be beneficial. It can be done by rotation, mirroring, and transitions of metal rails that do not hold the log. More time should be dedicated for the hyperparameter tuning. The number of nearest neighbors computed for each point, the number and parameters of subsampling and upsampling layers may affect the accuracy.

8 CONCLUSION

This thesis focuses on the problem of point cloud filtering from the noise that occurs during log scanning in the sawmill environment. The solution for this problem was proposed in the previous study, but it was not optimal for the industrial application. This work explores deep learning methods that solve this problem more robustly. The following work was done. The definition of deep learning was given, and some of the most popular architectures were described. A review of existing point cloud segmentation algorithms was made. The existing implementation of RandLA-Net was modified to work with the provided data. The method was trained and evaluated. It was shown that it is possible to automatically remove noise while maintaining a high level of accuracy. In future research, accuracy can be improved by training neural networks designed for semi-supervised learning.

REFERENCES

- [1] Forest Industries - Business Finland. <https://www.businessfinland.fi/en/do-business-with-finland/explore-key-industries/bio-circular-economy/forest-industries/forest-industries-in-brief>, 2022. [Online; accessed January, 22, 2022].
- [2] INDUSTRY | Sahateollisuus. <https://sahateollisuus.com/industry/?lang=en>, 2022. [Online; accessed January, 22, 2022].
- [3] Fedor Zolotarev, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, Tapio Helin, Heikki Haario, Tomi Kauppi, and Jere Heikkinen. Modelling internal knot distribution using external log features. *Computers and Electronics in Agriculture*, 179:105795, 2020.
- [4] DigiSaw - Leap of Digitalisation for Sawmill Industry. <http://www2.it.lut.fi/project/digisaw/index.shtml>, 2022. [Online; accessed January, 17, 2022].
- [5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.
- [6] MPAC Market Valuation Report Sawmills. <https://www.mpac.ca/sites/default/files/docs/pdf/Sawmills.pdf>, 2015. [Online; accessed January, 22, 2022].
- [7] Shankar Adhikari and Barbara Ozarska. Minimizing environmental impacts of timber products through the production process "From Sawmill to Final Products". *Environmental Systems Research*, 7:6, 2018.
- [8] Sahateollisuuskirja – Sahateollisuus-kirjan verkkomateriaali, 2022. [Online; accessed January, 18, 2022].
- [9] Michael Morin, Jonathan Gaudreault, Edith Brotherton, Frédéric Paradis, Amélie Rolland, Jean Wery, and François Laviolette. Machine learning-based models of sawmills for better wood allocation planning. *International Journal of Production Economics*, 222:107508, 2020.
- [10] Nicolás Vanzetti, Gabriela Corsano, and Jorge M. Montagna. Integrated scheduling of the drying process in a sawmill. *Computers & Chemical Engineering*, 153:107407, 2021.
- [11] A. Khouya and A. Draoui. Computational drying model for solar kiln with latent heat energy storage: Case studies of thermal application. *Renewable Energy*, 130:796–813, 2019.

- [12] Alfonso Lobos and Jorge R. Vera. Intertemporal stochastic sawmill planning: Modeling and managerial insights. *Computers & Industrial Engineering*, 95:53–63, 2016.
- [13] Fedor Zolotarev, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, Heikki Haario, Jere Heikkinen, and Tomi Kauppi. Timber Tracing with Multimodal Encoder-Decoder Networks. In *Computer Analysis of Images and Patterns*, pages 342–353, Cham, 2019. Springer International Publishing.
- [14] Dmitrii Shustrov, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, and Heikki Haario. Fine-Grained Wood Species Identification Using Convolutional Neural Networks. In *Image Analysis*, pages 67–77, Cham, 2019. Springer International Publishing.
- [15] Nikolay Rudakov, Tuomas Eerola, Lasse Lensu, Heikki Kälviäinen, and Heikki Haario. Detection of Mechanical Damages in Sawn Timber Using Convolutional Neural Networks. In *Pattern Recognition*, pages 115–126, Cham, 2019. Springer International Publishing.
- [16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [17] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.
- [18] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4 edition, 2021.
- [19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
- [20] Machine Learning Crash Course | Google Developers. <https://developers.google.com/machine-learning/crash-course>, 2022. [Online; accessed January, 25, 2022].
- [21] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Benamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4338–4364, 2021.
- [22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems*, 2017.

- [23] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.
- [24] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3D Recurrent Neural Networks with Context Fusion for Point Cloud Semantic Segmentation. In *European Conference on Computer Vision*, pages 403–417, 2018.
- [25] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018.
- [26] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian Yi Zhou. Tangent Convolutions for Dense Prediction in 3D. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018.
- [27] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet ++: Fast and Accurate LiDAR Semantic Segmentation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4213–4220, 2019.
- [28] Dario Reithage, Johanna Wald, Jurgen Sturm, Nassir Navab, and Federico Tombari. Fully-Convolutional Point Networks for Large-Scale Point Clouds. In *European Conference on Computer Vision*, 2018.
- [29] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. VV-Net: Voxel VAE Net With Group Convolutions for Point Cloud Segmentation. In *IEEE/CVF International Conference on Computer Vision*, 2019.
- [30] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse Lattice Networks for Point Cloud Processing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018.
- [31] Radu Alexandru Rosu, Peer Schütt, Jan Quenzel, and Sven Behnke. LatticeNet: fast spatio-temporal point cloud segmentation using permutohedral lattices. *Autonomous Robots*, 46(1):45–60, 2022.
- [32] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11105–11114, 2020.

- [33] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019.
- [34] Lin-Zhuo Chen, Xuan-Yi Li, Deng-Ping Fan, Kai Wang, Shao-Ping Lu, and Ming-Ming Cheng. LSA-Net: Feature Learning on Point Sets by Local Spatial Aware Layer. *arXiv preprint arXiv:1905.05442*, 2019.
- [35] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics*, 38(5), 2019.
- [36] Binh Son Hua, Minh Khoi Tran, and Sai Kit Yeung. Pointwise Convolutional Neural Networks. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018.
- [37] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *IEEE/CVF International Conference on Computer Vision*, pages 6410–6419, 2019.
- [38] Jiacheng Wei, Guosheng Lin, Kim Hui Yap, Tzu Yi Hung, and Lihua Xie. Multi-Path Region Mining for Weakly Supervised 3D Semantic Segmentation on Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4383–4392, 2020.
- [39] Xun Xu and Gim Hee Lee. Weakly supervised semantic point cloud segmentation: Towards 10× fewer labels. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13703–13712, 2020.
- [40] Hung Yueh Chiang, Yen Liang Lin, Yueh Cheng Liu, and Winston H. Hsu. A Unified Point-Based Framework for 3D Segmentation. In *International Conference on 3D Vision*, pages 155–163, 2019.
- [41] Maximilian Jaritz, Jiayuan Gu, and Hao Su. Multi-View PointNet for 3D Scene Understanding. In *IEEE/CVF International Conference on Computer Vision Workshop*, pages 3995–4003, 2019.
- [42] RandLA-Net in Tensorflow. <https://github.com/QingyongHu/RandLA-Net>, 2020. [Online; accessed June, 17, 2022].