# POLITECNICO DI TORINO

Master's Degree in Computer Engineering


Master's Degree Thesis

# Semantic Segmentation of Point Clouds for Urban Mapping: presentation and benchmarking of Turin3D Dataset

Supervisor

Prof. Paolo GARZA

Co-Supervisor

Dott. Luca BARCO

Dott. Giacomo BLANCO

Dott. Gaetano CHIRIACO

Candidate

Alessia INTINI

A.A. 2024/2025

**Abstract**

In recent years, considerable interest has emerged in improving urban planning and management, which has led to the need to explore new technologies. In particular, there has been a focus on the use of aerial aircraft for data acquisition in the urban environment. Although it is possible to acquire a large amount of data through these techniques, these are usually collected by LiDAR sensors, a technology widely used in applications using 3D data. The real problem is how to process and interpret them to extract useful knowledge in application contexts, so the use of these data collected by LiDAR sensors has led to the opening of new research scenarios in this field. The aim of this thesis is therefore to address the segmentation challenges inherent to the processing of 3D data acquired by aircraft operating in urban contexts, specifically obtain good results by testing different models on a new dataset containing data collected on a limited area of the city of Turin (about $1.43km^2$, divided into 57 blocks of about $25{,}000m^2$ each). The data were collected by a LiDAR sensor during an aerial flight on 29 January 2022 and comprise 69,591,759 points with a resolution of approximately 51.05 points per square meter. In order to use the dataset during the thesis experiments, it was necessary to divide it into three parts comprising the training (70%), test (20%) and validation (10%) subsets and also to manually label the test and validation to be able to obtain quantitative evaluations of the different models. Initially, the current state of the art in this field was evaluated by collecting public datasets and analysing the best architectures used in this area. With regard to datasets, those existing in the literature and presenting a taxonomy similar to that of Turin were selected, i.e. Sensat Urban, DELFT SUM, ECLAIR, FRACTAL, Toronto-3D, STPLS3D, Swiss3D and Hessigheim. Next, several neural network architectures that emerged from the literature as the most promising in 3D semantic segmentation tasks were considered, i.e. RandLA-Net, Point Transformer, KPConv and SPConv. In the course of the research,several techniques were employed to improve the segmentation results on Turin. First, it started with the transfer learning technique, using for the training phase the concatenation of the previously selected datasets and employing all the different neural network architectures. After that, these models were tested on the Turin dataset to identify the best performing architecture, subsequent experiments, in fact, only focused on the one that had performed best in this initial step. Subsequently, semi-supervised techniques were employed, using the confidence score of the best performing model's predictions on the Turin dataset as pseudo-labels during training. Several variants of these experiments were conducted to obtain the most accurate model possible on this new dataset.

# Table of Contents

# List of Tables

# List of Figures

# Acronyms

**ASCII**

**American Standard Code for Information Interchange** - A character encoding standard that represents text in computers and electronic devices.

**FBX**

**Filmbox** - A proprietary file format developed by Autodesk for the exchange of 3D data.

**FLS**

**Faro Laser Scan** - A file format used to store 3D scan data acquired with FARO laser scanners.

**GPS**

**Global Positioning System** - A satellite-based navigation system that provides precise location and time information.

**INS**

**Inertial Navigation System** - A navigation system using accelerometers and gyroscopes to calculate position, velocity, and orientation without external signals.

**LAS**

**Lidar Data Exchange Format** - A standard binary format for storing LiDAR point cloud data, designed to manage 3D spatial information including intensity and classification.

**LiDAR**

**Light Detection and Ranging** - A remote sensing technology that uses laser pulses to measure distances and create 3D models of environments.

**mIoU**

**Mean Intersection over Union** - A metric used to evaluate semantic segmentation performance by computing the intersection over union between predicted and ground truth areas.

**PCD**

**Point Cloud Data** - A file format used to store 3D point clouds, commonly associated with the Point Cloud Library (PCL).

**PLY**

**Polygon File Format** - A file format designed to store 3D models based on polygons and point clouds, supporting attributes like color and surface normals.

**RGB**

**Red Green Blue** - A color model based on the combination of three primary colors (red, green, and blue) to represent digital images.

**CNN**

**Convolutional Neural Network** - A type of deep neural network particularly effective for processing structured grid data, such as images, by using convolutional layers to automatically learn spatial hierarchies of features.

**RNN**

**Recurrent Neural Network** - A class of artificial neural networks designed for processing sequential data by maintaining a memory of previous inputs in their hidden states.

**MLP**

**Multilayer Perceptron** - A type of feedforward artificial neural network consisting of multiple layers of neurons, commonly used for supervised learning tasks.

**KNN**

**k-Nearest Neighbors** - A simple machine learning algorithm used for classification and regression by finding the k closest data points in the training set to a given query point.

**CPE**

**Class Prediction Error** - A metric used to evaluate the performance of a model by calculating the error in class predictions made for each sample.

**xCPE**

**Extended Class Prediction Error** - An extended version of the Class Prediction Error, considering additional factors or complexities in classifying data.

**LocSE**

**Localization Semantic Error** - An error metric used to assess the precision of a model in identifying the spatial location of objects or classes within a 3D point cloud.

**SC**

**Spatial Consistency** - A measure of how well the predicted labels for a point cloud align with the spatial structure of the environment, maintaining continuity in neighboring points.

**SSC**

**Semi-Supervised Classification** - A machine learning approach that uses both labeled and unlabeled data for training, aiming to improve model performance when labeled data is scarce.

**Nir**

**Near-Infrared** - A region of the electromagnetic spectrum with wavelengths just longer than visible light, often used in remote sensing for applications like vegetation analysis.

**MLS**

**Mobile Laser Scanning system** - A mobile system that captures 3D data using laser scanning technology mounted on a moving platform, often used for mapping and surveying.

**ULS**

**Unmanned Laser Scanning system** - A laser scanning system mounted on an unmanned aerial vehicle (UAV), used to collect 3D point clouds for various surveying and mapping tasks.

**ALS**

**Airborne Laser Scanning system** - A laser scanning system mounted on aircraft to collect high-resolution 3D data of the Earth's surface, often used for large-scale topographic surveys.

**SAP**

**Synthetic Aerial Photogrammetry** - A technique that uses aerial photographs to create synthetic 3D models of the environment, often combined with LiDAR data for enhanced accuracy.

# Chapter 1

# Introduction

In recent years there has been a significant interest in improving urban planning and management. This has brought the need to explore advanced technologies for the acquisition and analysis of data that could be useful in these areas. In particular, the focus is on data acquisition in urban environments, which can be carried out using various platforms such as aerial aircraft, drones (UAVs), mobile mapping systems, and ground-based surveying techniques. These innovations have opened up several new possibilities for monitoring cities, thus making it possible to collect information regarding urban morphology and apply it in various fields. For example, in urban planning, these data can be used to make various improvements, such as optimizing urban expansion, enhancing transportation networks, improving the allocation of green spaces, and many others.
Although it is possible to acquire a large amount of data through these techniques, these are usually collected by LiDAR sensors, a technology widely used in applications using 3D data. The real problem is how to process and interpret them to extract useful knowledge in application contexts, so the use of these data collected by LiDAR sensors has led to the opening of new research scenarios in this field.

The goal of this thesis is to address the segmentation challenges inherent in processing 3D data acquired from aircraft that operate in urban contexts. Therefore, in the course of this work, advanced segmentation techniques were looked for to extract relevant information from 3D point clouds. This required the use of robust segmentation models capable of classifying the various urban elements found in cities, such as buildings, streets, vegetation and other structures. Specifically, the identified models were tested on a new dataset, Turin3D, containing data collected over a limited area of the city of Turin.

## 1.1 Formulation of the problem

The data that are collected with LiDAR sensors constitute a great source of information, which can come in handy in countless fields, such as urban mapping for 3D city modeling, urban planning for sustainable development, and infrastructure management for road networks and public services. Through the collection of these data, it is learned how convenient it is to work with 3D rather than 2D data, as they are in fact more representative of reality. One of the most relevant aspects in this context is *3D Semantic Segmentation*, that is, the classification of each point, in a 3D point cloud, by determining its membership in a precise class. However, despite recent progress in deep learning applied to these 3D data, segmentation in the urban context still presents countless challenges, making it necessary to find more efficient solutions.

In the case of 3D data, the difficulty compared to 2D data lies in the fact that the data are unstructured and scattered, also having variable point distributions. In addition, especially in the urban context, several problems may be encountered that could worsen the performance of the models
The main ones can be summarized as :

- Significant variation in data density depending on the acquisition methods that are used, as LiDAR from aircraft or terrestrial.
- Problems of noise and occlusion, as shadow areas could be created in urban settings that make acquisition difficult.
- The high variability of urban architecture, in fact according to cities this can be very variable, just think for example of the presence or absence of rivers.
- Class imbalance, some will be much more present than others making it difficult for deep learning models to learn poorly represented classes.

All these issues make it necessary to look for advanced approaches.

## 1.2 Contributions

The thesis work follows several key steps. First, a comprehensive review of the state of the art in 3D data segmentation techniques is conducted, including the collection of public datasets, the analysis of the most effective architectures in this domain and also existing methodologies that can bring improvements. The next step involved

creating a new dataset, Turin3D, and annotating the test and validation subsets. Specifically, the dataset was divided into 70% for training, 20% for testing, and 10% for validation. This division was crucial for obtaining quantitative metrics to evaluate the performance of different models. Based on the Turin3D dataset, existing datasets with a taxonomy as similar as possible were selected. Additionally, the most promising neural network architectures for 3D semantic segmentation were identified through literature review. The experimental phase begins with transfer learning to determine the most suitable neural network architecture, followed by semi-supervised that using the confidence score of the best performing model's predictions on the Turin3D dataset as pseudo-labels during training. Quantitative metrics are used to actually evaluate how the model improved.

This thesis presents several innovative contributions aimed at advancing the field of 3D semantic segmentation, particularly in the context of urban environments. The work addresses key challenges, including the scarcity of annotated data and the need for more effective segmentation models. The main contributions of this research are summarized as follows:

- **Creation of the Turin3D Dataset**. The first major contribution is the creation of a a new dataset (Turin3D) which consists of 3D LiDAR point cloud data collected in the San Salvario neighborhood of Turin, Italy (it covers about $1.43km^2$, divided into 57 blocks of about $25{,}000m^2$ each). This dataset supports the segmentation of urban scenes and provides a new resource to the literature.

- **Transfer Learning** . Given the limited availability of labeled data, transfer learning was used to adapt pre-trained models on datasets in the literature to the Turin3D dataset. This approach allows knowledge from datasets with a similar taxonomy to be leveraged, addressing the challenges posed by the absence of annotated data and improving model performance.

- **Semi-Supervised Learning**. A semi-supervised learning approach was used to improve model performance by exploiting confidence scores on the training subset given by the best-performing transfer learning model. This method involves generating pseudo-labels for unlabeled data and applying class-specific confidence thresholds to ensure effective filtering of pseudo-labels, especially for underrepresented classes. An iterative refinement process was implemented, in which the model was retrained and updated pseudo-labels were progressively refined. Next, dynamic confidence thresholds were introduced to adaptively adjust the selection of pseudo-labels during each iteration, ensuring more accurate labels. Finally, the loss function was adapted to incorporate confidence scores, prioritizing the most reliable predictions and improving

the robustness, generalization, and overall segmentation performance of the model.

## 1.3 Outline of the work

In the thesis, the various topics will be addressed and the research will be divided into 6 chapters, here to define a brief description of each:

- **Chapter 1 - Introduction**. This chapter introduces the general problem addressed in this thesis and provides a formulation of the problem. The remainder of the chapter is structured as follows.
- **Chapter 2 - Related Works** . This chapter presents a review of existing research in the field of 3D semantic segmentation, analyzing state-of-the-art methods and solutions. This chapter helps identify the most effective approaches for addressing the research problem.
- **Chapter 3 - Datasets**. This chapter presents the datasets used in this research, including data collected in the city of Turin, the **Turin3D** dataset, and external datasets adapted to its taxonomy. Additionally, it provides a detailed description of the acquisition and construction process of the new Turin3D dataset.
- **Chapter 4 - Methodology**. The chapter covers all the methodologies that were chosen to perform the experiments and why some of the choices that could lead to improvements. All technical choices are described, including the different neural network architectures, training strategies and dataset used.
- **Chapter 5 - Experimental Results**. This chapter presents the results of the methodologies mentioned in the previous chapter, with the support of tables and other visual elements that can support the analysis and interpretation of the results.
- **Chapter 6 - Conclusions**. The final chapter summarizes the key contributions and findings of the research. It also discusses potential future directions for advancing 3D semantic segmentation based on the results obtained.

# Chapter 2

# Related Works

This chapter provides a brief overview of key concepts and a review of the existing literature on 3D semantic segmentation and its main paradigms. It begins by introducing the types of data being analyzed and the existing detection methods. Additionally, it presents an introduction to the most important neural networks and datasets used in this field.

## 2.1 LiDAR Sensor

LiDAR is an active optical sensor that emits laser pulses to a target and measures the signal return time, allowing the distance between the sensor and the object to be calculated. By combining these measurements with GPS and INS (Inertial Navigation System) data, LiDAR enables the reconstruction of detailed 3D point clouds of the environment. This technology is widely used in applications such as point cloud segmentation.
One challenge that can be found when using LiDAR sensors is in the detection of water surfaces. Due to the highly absorptive and reflective properties of water, LiDAR pulses often do not return meaningful signals, resulting in missing or very sparse points in point clouds. In addition, variations in water turbidity, surface motion and environmental conditions can further affect the accuracy and completeness of the acquired data. These limitations make water segmentation particularly challenging and often require additional processing techniques or complementary data sources to improve detection and classification.
Despite this LiDAR sensors can be used in various mapping and monitoring tasks because they can be mounted on aircraft, vehicles or tripods.

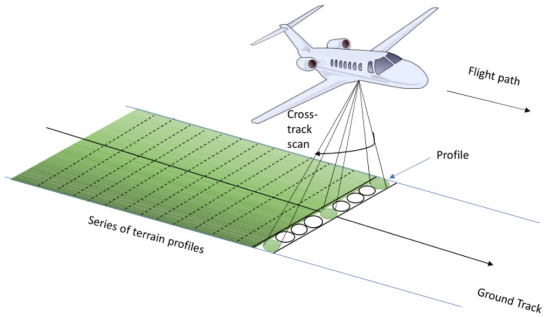



**Figure 2.1:** One of the possible vehicles that can perform LiDAR measurements. The image is recreated from figure 1.5 (b), pp. 8 in [1]. The figure is used for the first time in one of the authors' paper (see [2].)

Each recorded LiDAR point reports essential attributes, including intensity (signal strength), return number (indicating whether it is the first, second, or subsequent return from a single laser pulse), total number of returns, potentially RGB values, GPS timestamp, scan angle, and scan direction.
After LiDAR data are collected, post-processing becomes necessary, which requires the use of specific, often expensive and inaccessible software to perform. This limitation highlights the growing need for machine learning algorithms to automate classification, improving accuracy and reducing costs in large-scale LiDAR data analysis. [3]

## 2.2 3D Point Clouds

As mentioned earlier, in recent years, it has become necessary to work with 3D scenes and no longer 2D scenes since they are more representative of the real world. However, there are several possibilities for representing 3D data such as **point clouds**, **meshes**, and **voxels**, differences are shown in the figure 2.2.

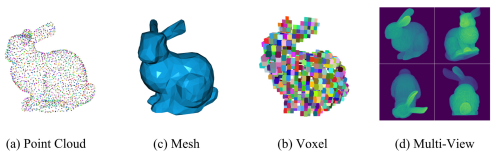



**Figure 2.2:** 3D data representation, the figure is used in [4].

The major differences between these are:

- **Point cloud**. It is a large collection of points shown in 3D space, these points can express the spatial distribution and surface characteristics of the target. Each point can contain many information, such as: three-dimensional coordinates (x, y, z), color information (r, g, b) and surface normal vector, etc.
- **Mesh**. Data are represented by a mesh grid and it's possible can be displayed as a collection of points that build local relationships between them.
- **Voxel**. Data are divided into a regular grid, these are useful in contexts where one wants to represent unevenly filled and regularly sampled spaces.
- **Multi-view**. In this technique, 3D objects are represented through images acquired from multiple angles.

In the current research there has been a focus on the representation of 3D data as point clouds, in fact these are one of the most popular forms of representation due to their ability to capture in detail the spatial distribution and geometric characteristics of objects.

There are various formats that can be used when it concerns 3D point cloud files; the main difference between these files is the use of **ASCII** and **binary**. The most common binary point cloud formats are FLS, PCD, LAS, etc. While other file formats that support both ASCII and binary are PLY, FBX [4]. In particular, the LAS and PLY formats are analyzed, which will be the ones mainly used in the case of the datasets under study. In the case of **LAS**, this is particularly popular for LiDAR data because of its ability to retain the detailed information they provide us, while in the case of **PLY**, they are designed for the representation of three-dimensional models and are able to support both geometric information and attributes such as color and transparency.

## 2.3 3D Semantic Segmentation

Semantic segmentation of 3D point clouds in urban environments is a key task in urban mapping and scene understanding, enabling the classification of each point into a specific semantic category. This process is essential for applications such as city modeling, infrastructure monitoring, and autonomous navigation. To achieve this, existing methods can be categorized into four main paradigms: **Projection-based**, **Discretization-based**, **Point-based**, and **Hybrid methods** [5]. In the case of the first two paradigms, before the other steps, they transform the point cloud into an intermediate regular representation, then the intermediate segmentation results are projected onto the raw point cloud. In contrast, point-based methods can work on irregular point clouds without performing any transformation.

The following methods focus on point-based semantic segmentation and hybrid methods. In the former, one directly works on the raw point cloud without requiring any transformation, while in the latter they combine point-based techniques with other representations, such as voxel grids, meshes or images, to improve the accuracy and efficiency of segmentation.

## 2.4 Point-based Neural Networks

As mentioned above, if it is in the context of point-based methods, the networks work directly with irregular point clouds. The main problem is that the raw point clouds acquired by depth sensors are typically irregularly sampled and therefore do not have a regular structure [6]; this consequently leads to the impossibility of applying traditional CNNs [7]. For this end, PointNet [7] introduced shared MLPs for point-based feature learning and a symmetrical pooling function for global feature learning [5].

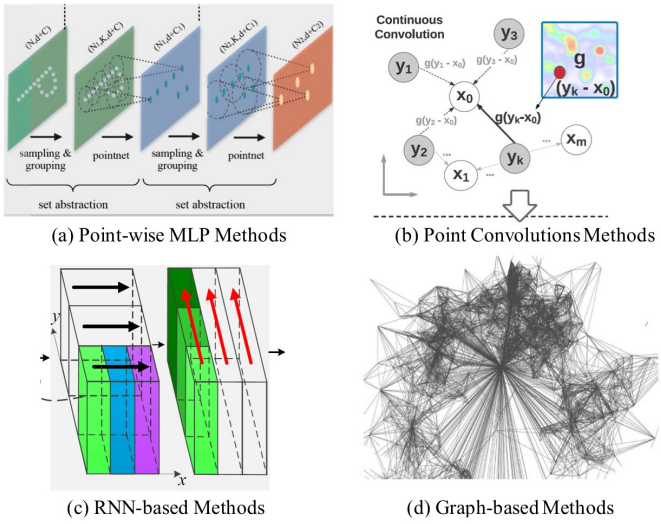



**Figure 2.3:** An illustration of point-based methods, shown in [5]

Several PointNet-based networks have been proposed in recent years, which can be categorized into four main groups:

- **Pointwise MLP Methods**. Use shared MLPs, however these are not able to capture the local geometry and mutual interaction between points [7]. For this reason, new techniques have been introduced such as methods based on neighbouring feature pooling, attention-based aggregation, and local-global feature concatenation [5].

- **Point Convolution Methods**. Use convolution operators for point clouds; Hua et al. [8] proposed a point-wise convolution operator in which neighbouring points are divided into kernel cells and then convolved with kernel weights. For example, in the case of KPConv, the convolution weights of this are determined by the Euclidean distances to kernel points, and the number of kernel points is not fixed.

- **RNN-based Methods**. It employs RNNs to capture the spatial dependence between points; experiments show how integrating spatial context can improve segmentation performance [5].

- **Graph-based Methods**. Represent point clouds by means of graphs to model geometric relationships [5].

Based on these methods, it is possible to explore different types of neural networks that have different characteristics and strengths.

### 2.4.1 PointNet

PointNet [7] was introduced to address the limitations of traditional convolutional architectures, which typically require highly regular input data formats. While such formats facilitate convolution operations, they also lead to unnecessarily large data volumes and introduce quantization artifacts, which can obscure the natural invariances of the data. To overcome these issues, PointNet directly processes raw point clouds, assigning labels to each point without requiring data transformations.

In this architecture, all points are processed identically and independently, its main feature being the use of a single symmetric function, max pooling. This allows the network to learn a set of optimal criteria that select interesting points and encode the reason for their selection, while the final, fully connected layers of the network aggregate these optimal values and serve to predict the labels for each point.

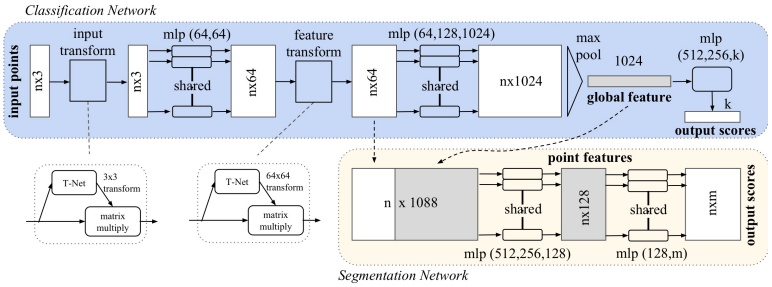



**Figure 2.4:** PointNet[7]: Classifies n points via MLPs, feature transformations, and max pooling. Segmentation extends it by combining global and local features. Uses BatchNorm, ReLU, and Dropout.

The structure of the network is illustrated in Figure 2.4, where *the classification network* and *the segmentation network* share most of their architecture. The classification network takes *n points* as input, applies input and feature transformations, and then aggregates the point features using max pooling, producing classification scores for k classes. The segmentation network can be seen as an extension of the classification network, where global and local features are concatenated to generate per-point classification scores.

The PointNet architecture is based on three key modules:

1. **Maximum global pooling as a symmetrical feature**. Aggregates point information and eliminates dependency on input order.
2. **Combination of local and global information**. After obtaining the global features, they must be concatenated with the local features of each point to improve segmentation.
3. **Alignment networks with the T-Net**. This is a module that learns spatial transformations to align the point cloud in a canonical space, this guarantees robustness to geometric variations.

PointNet[7] underlies all methods that directly process 3D point clouds and learns features on a per-point basis using shared multilayer perceptrons. While this approach is computationally efficient, it is unable to capture broader context information for each point. For this reason, PointNet was the basis for other networks developed later that aimed to improve on its limitations.

### 2.4.2 RandLA-Net

As explained earlier Pointnet[7] had limitations that its successors tried to improve, for this reason, many dedicated neural modules were subsequently and rapidly introduced. These methods can be divided into four categories: *Neighbouring Feature*

*Pooling*, *Graph Message Passing*, *Kernel-based Convolution* and *Attention-based Aggregation*. These can be efficient in the context of semantic segmentation, but most of them are limited to use in scenarios where the point clouds are small. This limitation stems from three main factors: commonly used sampling methods are computationally and memory expensive, local feature learning algorithms are computationally expensive, and existing networks have limited receptive fields, making them inefficient for capturing complex structures in large point clouds.

This led to the introduction of **RandLA-Net**[6], an efficient neural architecture capable of directly processing large-scale 3D point clouds without requiring any pre/post-processing steps. The key concept behind RandLA-Net is *Random Sampling*, which is the most efficient way to process large-scale point clouds. In fact, this significantly reduces the point density while simultaneously applying a carefully designed local feature aggregator to preserve the most relevant features; this approach achieves an optimal balance between efficiency and effectiveness.

To understand Random Sampling, its key characteristics can be outlined as follows: it uniformly selects *K points* from *the original N points*, with a computational complexity of $\mathcal{O}(1)$. Compared to other methods, it has the highest computational efficiency, regardless of the scale of the input point clouds and it does not require extra memory for computation. The problem is that this may involve the removal of many useful point features, to overcome this problem, RandLA-Net introduces a new local feature aggregation module.

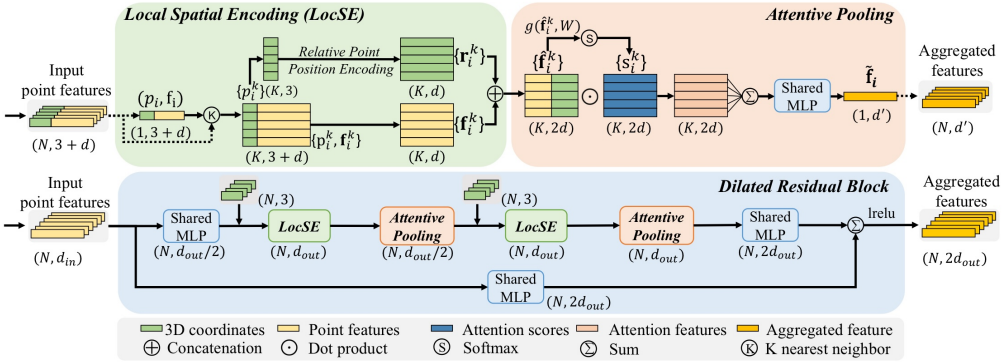



**Figure 2.5:** The proposed local feature aggregation module. The top panel shows the location spatial encoding block while the bottom panel shows how two of these components are chained together, to increase the receptive field size, within a residual block [6].

As can be seen in the figure 2.5, there are three different neural units:

- **Local Spatial Encoding (LocSE)**. Its task is to encode the relative spatial coordinates of neighbouring points and in this way improve the network's

ability to learn local geometric structures. To make this possible, it uses the K-Nearest Neighbours (KNN) method that finds neighbouring points and an MLP function to calculate relative representations. By combining this information with the features of the neighbouring points, an enriched feature set is generated.

- **Attentive Pooling**. Its purpose is to aggregate the features of neighbouring points using an attention mechanism, so that there is no lost information. It calculates attention scores for each feature and then uses a weighted sum to select the most relevant information.

- **Dilated Residual Block**. It expands the receptive field by combining multiple layers of LocSE and Attentive Pooling with residual connections, inspired by ResNet[9] and dilated networks[10] . In this way, each point can acquire information from an increasing number of neighbours, improving its ability to capture larger structures but limiting the increase in computational cost.

### 2.4.3 Point Transformer V1 Network (PTv1)

Other possible architectures to consider could be those based on transformers. The introduction of transformers has brought significant advantages, and **PointTransformer** [11] extends these innovations to the field of 3D point cloud processing. This neural network architecture exploits the self-attention mechanism, which operates without a fixed structure. Unlike other networks that voxelize 3D space or apply convolutions on graphs, PointTransformer uses local attention based on a variant of vector self-attention combined with efficient positional coding. This approach allows the model to accurately capture the geometric structure of 3D scenes.

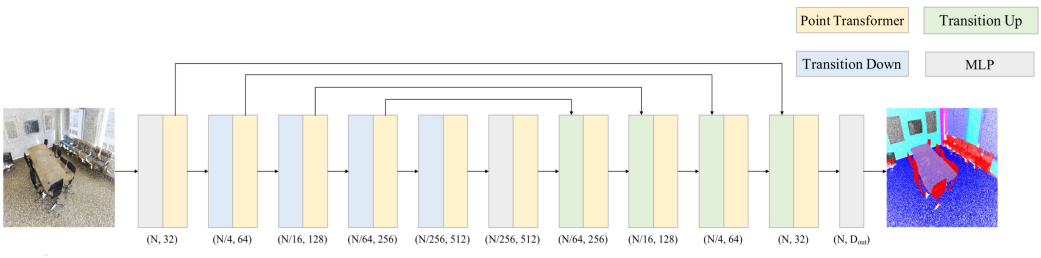



**Figure 2.6:** Point transformer[11] network for semantic segmentation.

Figure 2.6 shows the structure of the network. At first, the **Residual Point Transformer block** can be described as using a self-attention layer as its central element, combined with linear projections to reduce dimensionality and a residual connection to facilitate information flow. Using this block, the network adopts an encoder-decoder architecture, comprising five downsampling stages that progressively reduce the number of points. Consecutive stages are connected by transition modules: the **transition down** for feature encoding and the **transition up** for feature decoding. The transition down module employs farthest point sampling to select a well-distributed subset of points and uses a kNN graph to aggregate local features through max pooling. On the other hand, the transition up module interpolates features to a higher resolution and combines them with the encoder's features via a skip connection. As for semantic segmentation, the final decoding stage generates a feature vector for each point in the input point set.

### 2.4.4 Kernel Point Convolution Network (KPConv)

Another existing architecture that works with 3D points is **KPConv**[12], which operates on point clouds without the need for intermediate representations such as voxels or regular grids. This neural network architecture is given its name by a new operation that is **Kernel Point Convolution (KPConv)**, which is a novel convolution to operate on 3D point clouds. This approach is distinguished by the use of a set of spatially distributed kernel points, which define the weights of the convolution, allowing greater flexibility than convolutions using fixed grids.
The most relevant feature of this operation is its deformable version, which allows the kernel points to dynamically adapt their positions based on the geometry of the scene. This capability is particularly useful in urban environments, where there is significant variation in shape and density. In addition to this adaptability, the use of radius neighborhoods instead of traditional k-nearest neighbors provides further advantages in the urban domain. Since datasets in this field are often acquired through LiDAR sensors, they may be sampled non-uniformly. In such cases, this technique can help improve performance.

### 2.4.5 Submanifold Sparse Convolutional Network

Another commonly used approach in general is based on Convolutional Networks. However, traditional Convolutional Networks are optimized for dense grid-based data, such as 2D images, and become inefficient when applied to sparse data due to their high computational cost. One of the main challenges when applying standard Convolutional Networks to 3D data is that each convolution operation progressively increases the number of active points in space, leading to a loss of the original sparsity. This issue, known as the Submanifold Dilation Problem, is addressed by

restricting the convolution output to only the active points present in the input. However, this approach may result in the separation of neighboring connected components, hindering the propagation of information.
To overcome these limitations, **Submanifold Sparse Convolutional Networks (SSCN)** [13] were introduced. These networks propose a new implementation for performing **Sparse Convolutions (SC)** and introduce a novel convolution operator, **Submanifold Sparse Convolution (SSC)**, specifically designed to maintain sparsity while preserving spatial continuity.

- **Sparse Convolutions (SC)**. It is a type of convolutional operation optimized for scattered data, such as point clouds. Unlike standard convolutions, SC assumes that non-active sites have zero value, reducing the computational cost.

- **Submanifold Sparse Convolution (SSC)**. It is a modified SC convolution, which preserves the sparsity structure of the point clouds. In an SSC operation, the output retains the same size as the input through appropriate padding, ensuring that a point in the output is active only if the corresponding point in the input is also active.

To construct this convolutional network based on SC and SSC, other operators are defined such as Activation and Batch Normalization functions, Max Pooling (MP) and Average Pooling (AP) and Deconvolution (DC).

The use of SSCs makes it possible to maintain high efficiency and accuracy in point cloud processing, avoiding the wastage of computational resources on empty regions.

## 2.5 Datasets

Several datasets exist to evaluate the performance of models using 3D point clouds. In particular, those reported here are datasets used for 3D semantic segmentation and representing urban environments; however, each of them has a different taxonomy. The difference between them beyond the taxonomy is the acquisition; in fact, it is possible to acquire these datasets from different types of sensors, including *Mobile Laser Scanners (MLS)*, *Aerial Laser Scanners (ALS)*, *static Terrestrial Laser Scanners (TLS)*, *RGBD cameras* and other 3D scanners [5]. They may also contain different fields for each point, depending on the type of sensor used to collect the data.

### 2.5.1 Sensat Urban

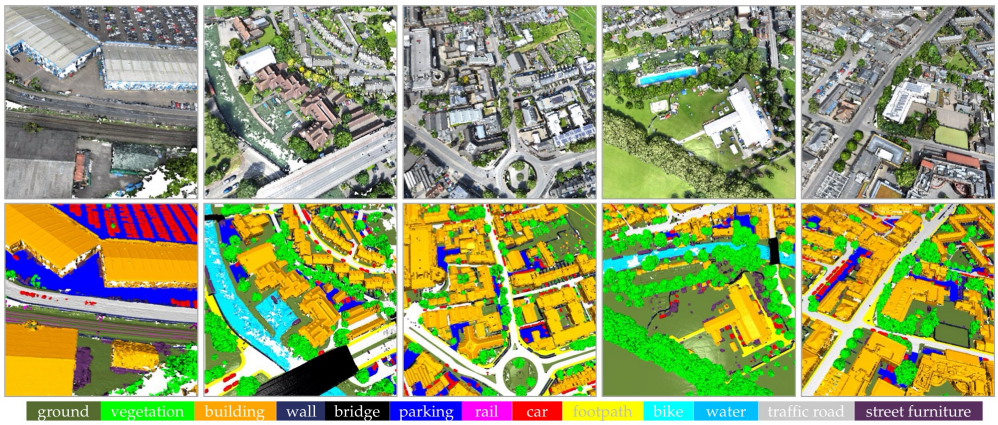



**Figure 2.7:** Examples of SensatUrban dataset, different semantic classes are labeled by different colors.[14]

**SensatUrban** [14] is an urban-scale photogrammetric point cloud dataset and containing almost three billion semantically annotated points. It covers approximately $7.6km^2$ of urban landscape in three UK cities: *Birmingham*, *Cambridge* and *York*. The data were collected through aerial surveys, with flight paths pre-planned in a grid pattern and automated using the e-Motion flight control system. To cover the entire area efficiently, multiple flights were conducted in parallel, each lasting 40 to 50 minutes. The collected data were then processed using commercial software such as Pix4D, which applies Structure from Motion (SfM) and dense image matching techniques for point cloud reconstruction.
In total, the dataset consists of: 569 million points over $1.2km^2$ that are part of the Birmingham suburban area, over $2.27billion$ points over $3.2km^2$ for the Cambridge urban area, and finally about 904 million points over $3.2km^2$ for York. However, the only labeled data are those for Birmingham and Cambridge, these follow the below split: 14 tiles for Birmingham (10 for training, 2 for validation, and 2 for testing) and 29 for Cambridge (20 for training, 5 for validation, and 4 for testing), each tile is about $400X400m^2$. However, the creators of the dataset have not released the labeled data for the test subset; only the labels for the other subsets are available. The labels used in the dataset follow 13 semantic classes, the main ones of which are: *Soil (0)*, *Vegetation (1)*, *Buildings (2)*, *Cars (9)*, *Water (12)*, and so on; an example of how the labeling is done can be found in Figure 2.7.
SensatUrban represents a significant contributor to large-scale applications, including urban planning and infrastructure management.

### 2.5.2 SUM

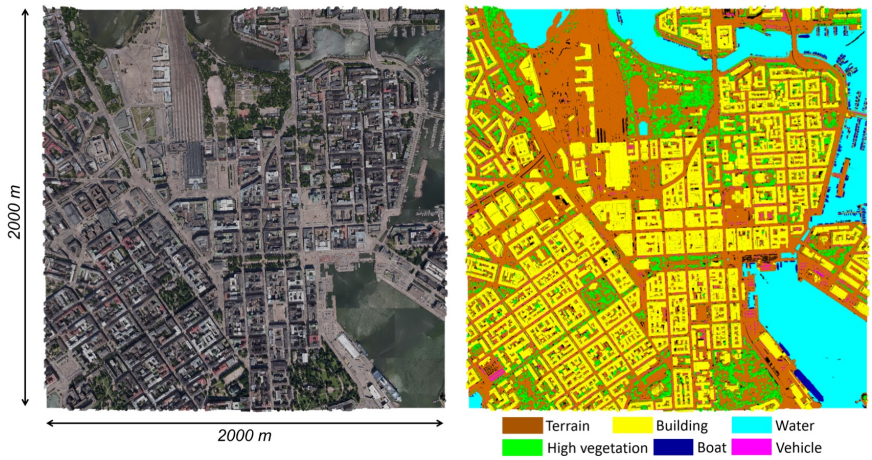



**Figure 2.8:** Overview of the urban mesh in SUM. Left: textured meshes covering an area of approximately $4km^2$. Right: ground truth meshes.[15]

The **SUM**[15] dataset is a high-resolution 3D urban dataset covering approximately $4km^2$ in the centre of Helsinki, Finland.
It consists of a semantically annotated triangular mesh with six classes: *Terrain (1)*, *Building (3)*, *High Vegetation (2)*, *Water (4)*, *Vehicle(5)* and *Boat(6)*. The mesh was generated from oblique aerial images with a GSD of 7.5 cm, acquired in 2017 using an aircraft-mounted multi-camera system, while reconstruction was performed using with ContextCapture software, which applies techniques of aerial triangulation, dense image matching and surface reconstruction.
This dataset is an important resource for urban semantic segmentation due to its extent and accuracy, offering high quality data and detailed annotation.

### 2.5.3 Eclair

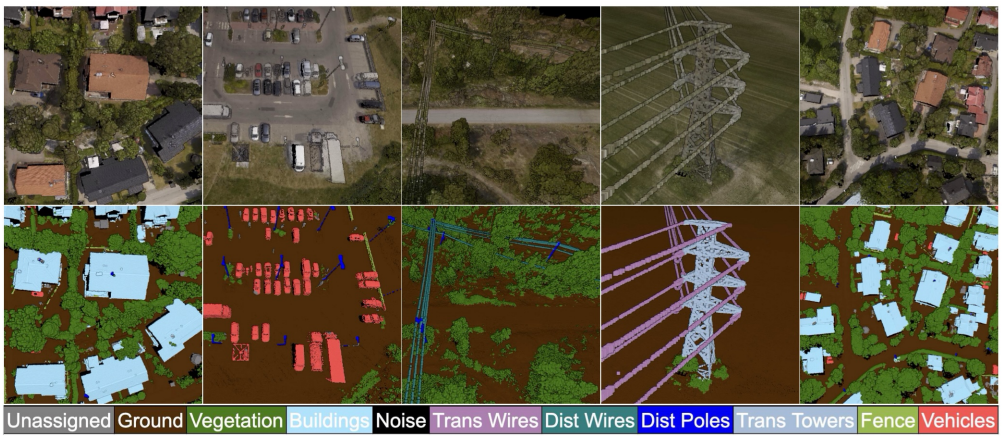



**Figure 2.9:** An example of the ECLAIR annotation, which covers $10km^2$ and contains 11 semantic classes[16].

The **ECLAIR** (Extended Classification of LiDAR for AI Recognition)[16] dataset is a large LiDAR dataset acquired in the city of Espoo, Finland. It covers an area of more than $10km^2$ and contains approximately 600 million points. This dataset was developed to support urban research, furthermore, in contrast to other terrestrial LiDAR datasets, ECLAIR was obtained using Airborne LiDAR Scanning (ALS). This method provides a wider and more uniform coverage than others, but with higher costs. The acquisition was performed with a proprietary multi-sensor system mounted on a helicopter and the data was collected at a flight height of 100 meters and speed of 40 knots. With 600 PRR (Pulse Repetition Rate) and 234.5 lines per second, the LiDAR has a point density of $50 points/m^2$ and a swath width of 328 meters.
After the data were collected, the dataset was processed by converting the raw data into LAZ format, subsequently subdividing the point clouds into 1246 tiles (with a size of 100 × 100 m ) for more efficient processing. In addition, the dataset was divided as follows: 70% of the data represents the training subset, 10% the validation subset and 20% the testing subset. This partitioning ensures a balanced evaluation of model performance across different stages of learning. Notably, the dataset comprises 11 semantic classes, with a predominant focus on electrical infrastructure, including categories such as *Transmission Wires (6)*, *Distribution Wires (7)*, *Poles (8)*, and *Transmission Towers (9)*, among others.

### 2.5.4 Fractal

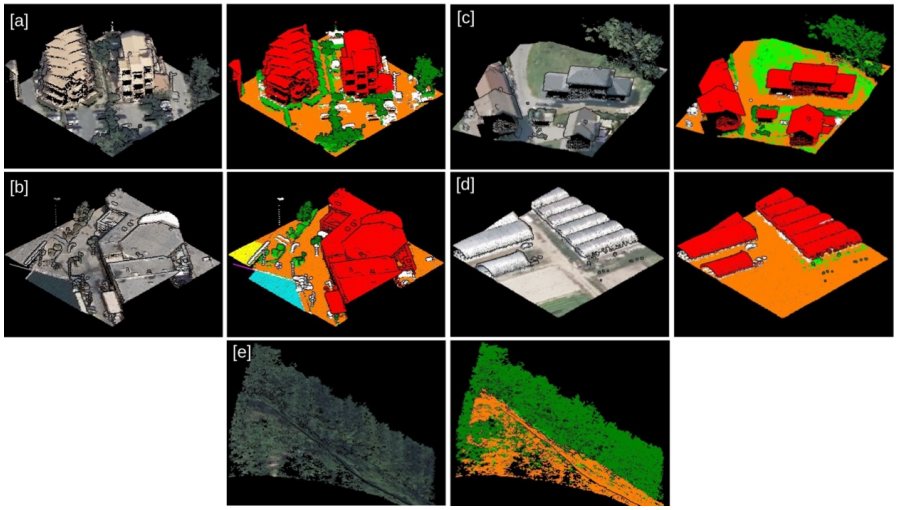



**Figure 2.10:** An example of an annotation for FRACTAL dataset, representing the different scenes it contains[17].

The **FRACTAL**(FRench ALS Clouds from TArgeted Landscapes)[17] dataset is a large-scale LiDAR dataset designed for the semantic 3D segmentation of heterogeneous landscapes. It consists of 100,000 point clouds acquired by Airborne LiDAR Scanning (ALS) and covers a total area of $250km^2$ selected from an initial area of $17{,}440km^2$ in five regions of France. This dataset was constructed using open-source data from *the Institut national de l'information géographique et forestière (IGN)* as part of the Lidar HD programme, which aims to achieve high-density 3D mapping of the entire French territory by 2026.
During the construction of the dataset, a sampling scheme is used that explicitly focuses on rare classes, rare objects and challenging landscapes. In general, FRACTAL includes 9261 million points with an average density of 37 points/$m^2$ and annotation was provided for 11 semantic classes which are: *Ground (2)* , *Vegetation (low, medium and high)*, *Buildings (6)*, *Water (9)*, *Bridge (17)*, *Permanent Structures (64)*, *Artefact (65)*, *Synthetic (66)*. Furthermore, the existing division for the dataset is as follows 80% for the training set, 10% for the validation set and 10% for the test set.

### 2.5.5 Toronto3D

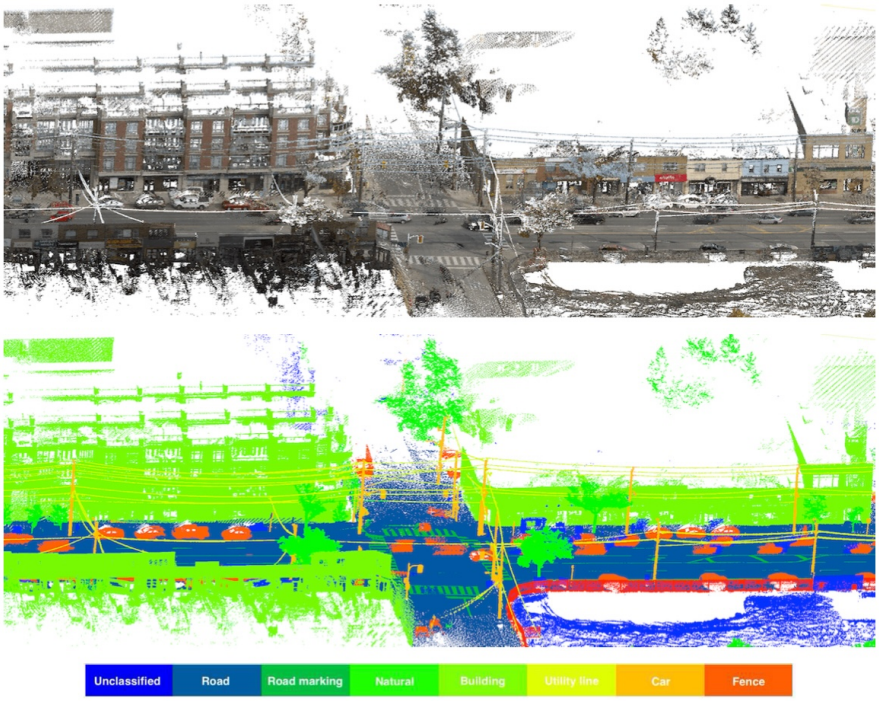



**Figure 2.11:** A section of the Toronto3D dataset is shown, with the top part displaying the dataset in natural colors (RGB) and the lower part illustrating its classification into 8 semantic classes[18].

**Toronto-3D**[18] is a large-scale urban outdoor point cloud dataset for 3D semantic segmentation of urban environments; it is acquired through a mobile laser scanning system (MLS) in Toronto, Canada and it covers $1km$ of urban streets comprising approximately 78.3 million points. In particular, the devices used to collect the data are a 32-line Teledyne Optech Maverick LiDAR sensor, integrated with a Ladybug 5 panoramic camera and a GNSS system for geolocation.
Also thanks to these technologies each point contains different information such as: *the location of each point recorded in meters (x,y,z)*, *natural color (R, G, B)*, *intensity*, *GPS time*, *scan angle* and *label*. The labels included in the dataset are manual annotations consisting of 8 classes, some of them are : *Road (1)*, *Road Markings (2)*, *Natural (3)*, *Building (4)*, *Utility Line (5)*, *Fence(8)*, etc. It can be observed that there are two class labels not commonly spread in other datasets, these classes are *road marking* and *utility line*.

### 2.5.6 STPLS3D

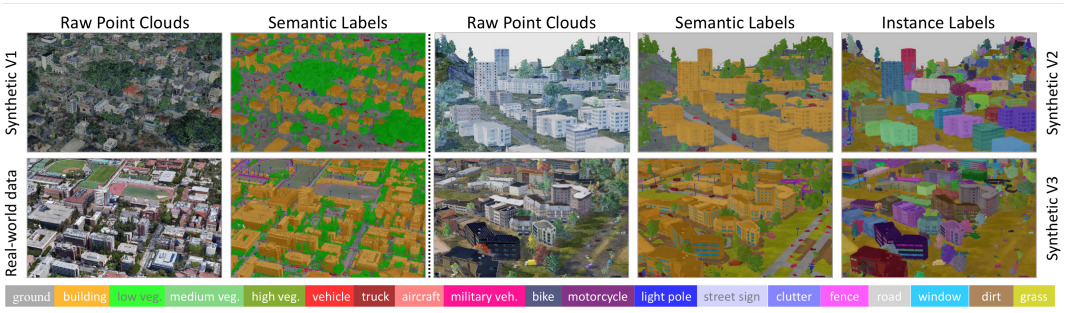



**Figure 2.12:** Examples of STPLS3D datasets, including Synthetic V1, Synthetic V2, Synthetic V3 and real-world subsets. The different semantic classes are shown below[19].

**STPLS3D**[19] is a large-scale dataset designed for semantic segmentation derived from aerial photogrammetry, covering more than $16km^2$ of landscapes and including 18 semantic categories. Some of these categories are: *Ground (0)*, $16km^2$, *Vegetation (low, medium, and high)*, *Truck (6)*, *Aircraft (7)*, *Military Vehicle (8)*, *Light Pole (11)*, *Street Sign (12)*, *Clutter (13)*, *Dirt (18)*, etc.
The novelty of this dataset compared to others is that it combines real-world data acquired from UAVs with three synthetic versions (V1, V2, and V3) generated through a procedural pipeline. This approach addresses common challenges in real-world data collection and annotation, such as class imbalance and heterogeneous point quality. Synthetic data were obtained through a generation pipeline that mimics the real photogrammetric acquisition process by simulating UAV flights over virtual environments. These environments were built using open geospatial data and procedural modeling tools, enabling the creation of realistic 3D point clouds that remain compatible with real-world data while eliminating the need for manual annotations.
In conclusion, the integration of synthetic data, as in this case, can significantly influence the efficiency and generalization of deep learning models, bridging the gap between real-world complexity and the need for large-scale annotated datasets.

### 2.5.7 Swiss3D

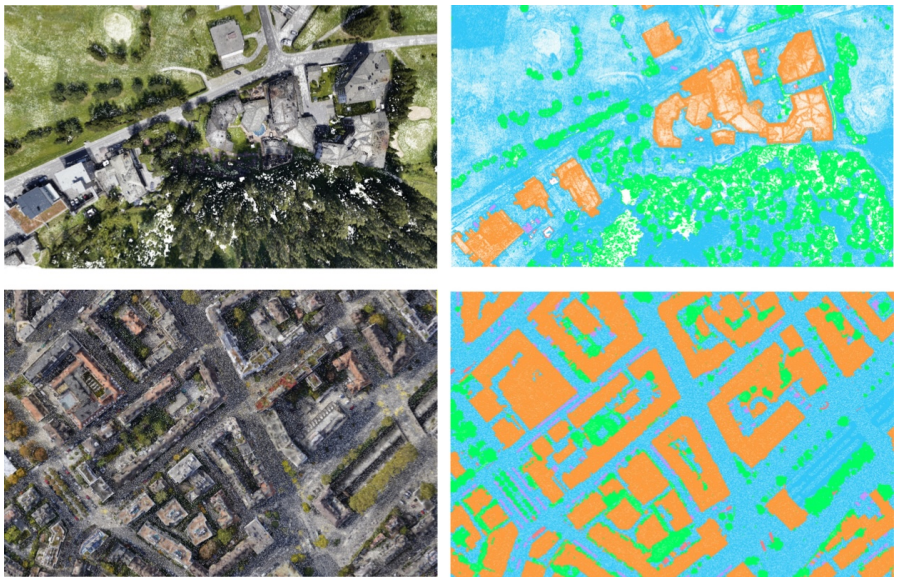


**Figure 2.13:** An example of Swiss3D dataset, on the left can be seen the version with natural colors(R,G,B) and on the right its classification[20].

**Swiss3DCities**[20] is a large-scale dataset designed for semantic segmentation of 3D point clouds acquired by aerial photogrammetry. The dataset covers a total area of $2.7km^2$ spread over three Swiss cities with different urban characteristics: *Zurich* (dense and urban), *Zug* (mixed, with industrial and residential areas), and *Davos* (mountainous and rural). The data as mentioned earlier were collected by multi-rotor drones following dual-grid flight paths, the main difference from data collected by LiDAR sensors is that denser and more complete point clouds are obtained.
The entire dataset is manually annotated with dense labels, the categories chosen for semantic segmentation labels are: *Terrain (1)*, *Building (2)*, *Urban Asset (3)*, *Vegetation (4)* and *Vehicle (5)*.
With its high-resolution acquisition and accurate semantic annotation, it represents one of the best datasets for developing advanced techniques in 3D semantic segmentation.

### 2.5.8 Hessigheim


(a) H3D(PC) - RGB as derived from H3D(Mesh)

(b) H3D(PC) - Class labels

(c) H3D(Mesh) - RGB

(d) H3D(Mesh) - Class labels transferred from H3D(PC)

**Figure 2.14:** An example of Hessigheim dataset, on the left can be seen the version with natural colors(R,G,B) and on the right its classification into 11 semantic classes[21].

**Hessigheim 3D**(H3D)[21] is a dataset used for semantic segmentation of 3D point clouds and textured meshes, data were acquired from a LiDAR system and cameras integrated on the same Unmanned Aerial Vehicle (UAV) platform. The data were collected in the village of Hessigheim, Germany, and in three different time periods: March 2018, November 2018, and March 2019 in order to be used in change detection applications.
A distinctive feature of this dataset is its high spatial resolution; in fact, for the point cloud there is a density of about 800 points/$m^2$, while the 3D mesh is textured with images that have a ground resolution of 2-3 cm. The point cloud was entirely annotated manually and later this annotation was transferred to the 3D mesh by a geometric association method. It is possible to observe 11 semantic classes included in the H3D annotation, some of these classes are: *Low Vegetation (0)*, *Impervious Surface (1)*, *Vehicle (2)*, *Urban Furniture (3)*, *Roof (4)*, *Shrub (6)*, *Vertical Surface (9)*, etc.

## 2.6 Transfer Learning

**Transfer learning**[22] is a crucial technique for improving the semantic segmentation of 3D point clouds, in particular in scenarios where labeled data is limited. Using this approach, knowledge learned on large, well-annotated datasets can be transferred to new domains that do not contain fully labeled data. In addition, a pre-processing pipeline must be used for efficient point cloud segmentation; this is necessary to ensure uniformity, compatibility and accuracy in the subsequent segmentation model. The key steps are:

- **Converting Batches to Same Size**. The same size is given to each batch, so that an equal number of points are generated for each. This uniformity facilitates a training process and guarantees that each batch is of consistent size, enabling efficient model convergence.

- **Class Remapping**. The datasets used may have different class labels, so a process of remapping the classes is necessary to align them with each other. This remapping process ensures consistency of class representation across datasets, contributing to a unified segmentation model.

As demonstrated in [22], this approach produces high-quality 3D point cloud segmentation while effectively addressing data sparsity issues, offering a robust solution for urban point cloud analysis.

## 2.7 Semi-Supervised

In 3D semantic segmentation, having large amounts of labeled data is crucial for training accurate models; however, the annotation process is expensive and complex. For this reason, the use of **Semi-Supervised Learning (SSL)** techniques is useful, as it allows unlabeled data to be used. In particular, *pseudo-labels*, e.g. based on the confidence of predictions, can be generated to improve model learning without depending exclusively on labeled data.

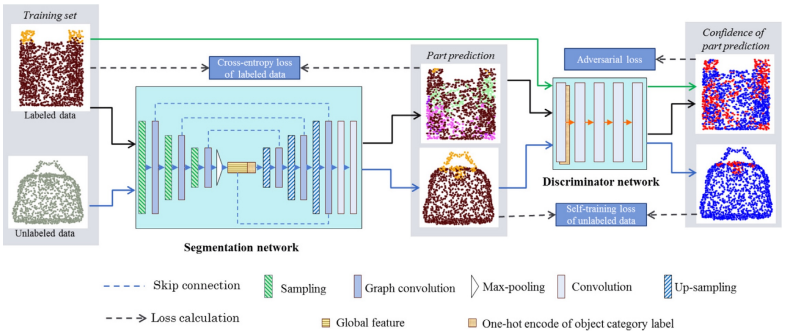



**Figure 2.15:** Overview of method described in [23]

There are several methods that can be used to implement these SSL techniques. One of these is the one described by *Hongyan Li et al. (2021)*[23], which proposes an approach based on self-learning and confidence evaluation of pseudo-labels. This method integrates self-training with generative adversary networks (GANs) to improve the reliability of pseudo-labels assigned to unlabeled data. The architecture consists of two networks: a segmentation network, which performs semantic segmentation, and a discrimination network, which evaluates the reliability of the generated labels, selecting only those that are considered highly reliable. It is possible to see an overview of the model used in the figure 2.15.
The specific process performed in this method is iterative:

1. The model is initially trained with labeled data only.

2. After an initial training phase, this model is used to assign pseudo-labels to unlabeled point clouds.

3. The discriminator evaluates the confidence of the labels and selects the reliable ones.

4. The selected pseudo-labels are added to the training set to improve the model.

Another method proposed by *Zhimin Chen et al.(2022)*[24] focuses on the selection of variable confidence thresholds for the choice of pseudo-labels. The use of fixed thresholds can in fact generate two main problems. **The first concerns the unbalancing of classes**: the most represented classes tend to dominate the learning process, while the least represented classes receive an insufficient number of pseudo-labels, compromising the effectiveness of the training. **The second issue is the poor adaptability of thresholds**, a fixed threshold does not take into account the different learning difficulties between classes, reducing the efficiency of using unlabeled data and limiting the improvement of the model. To overcome these limitations, the proposed method dynamically adapts the pseudo-label selection

threshold according to the average confidence of each class, optimizing the training process.

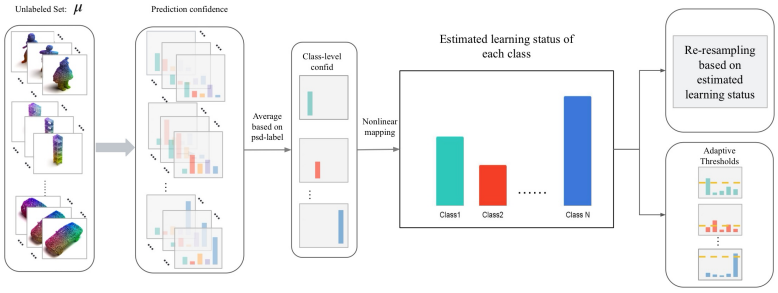



**Figure 2.16:** Overview of method described in [24], it consists of three main parts: (1) Determining the learning status of each class from unlabeled data, (2) dynamically adjusting class-specific thresholds, and (3) re-sampling the dataset accordingly.

The main strategies behind this method are:

1. **Dynamic Thresholding.** Instead of using a fixed threshold to decide whether to include a pseudo-label, the threshold is dynamically adjusted according to the average confidence of the class. For example, for classes with low confidence, lower thresholds are selected to include more labeled data.

2. **Dynamic Re-Sampling.** Even with dynamic thresholds, some classes may still dominate others; therefore, confidence is dynamically adjusted. At each iteration, the confidence of each class is monitored and updated to continuously improve the model.

The integration of these strategies optimizes the use of unlabeled data and improves the generalization of the model.

# Chapter 3

# Turin3D Dataset

This chapter will principally describe the Turin3D dataset, which was the main dataset used during the thesis work to evaluate the performance of the models and improve them. It will also explain how the classes of all the other datasets examined were mapped so that they were consistent with the taxonomy of the Turin3D dataset.

## 3.1 Dataset Description

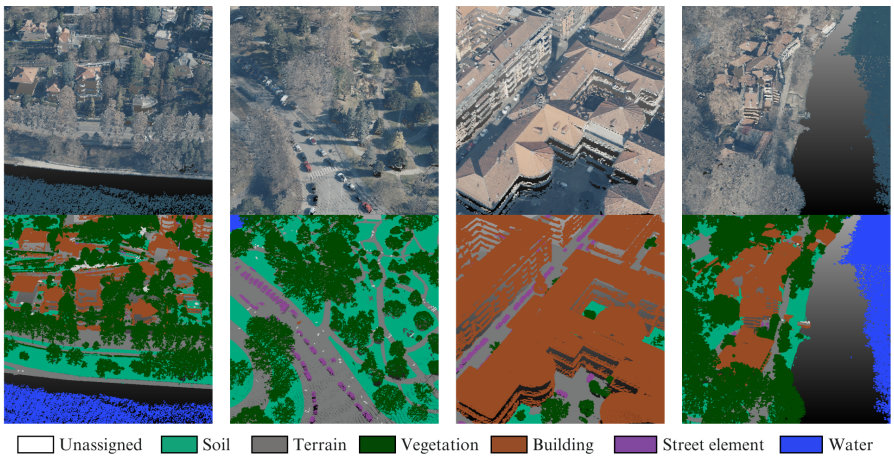



**Figure 3.1:** Overview of the Turin3D dataset, with an example of the annotation followed and the segmentation process.

This section will focus on the description of the new Turin3D dataset, acquired with LiDAR sensors in a limited area of Turin.
The current dataset was collected in the district of San Salvario in Turin, Italy,

and covers a total area of approximately $1.43km^2$; the entire area was subdivided into 57 blocks, each of which has a size of approximately $25{,}000m^2$. To better understand the spatial distribution of these points and the diversity of the captured environments Figure 3.2a illustrates the area covered by the dataset, revealing the variety of landscapes present. In fact, this area of the city includes several examples of urban environments, moving from green areas and parks to urban centres, and also including bridges and rivers.
The number of points per block varies significantly depending on the location of the point cloud. This variation is due to the diversity of environments within the selected area. As shown in Figure 3.2b, the left side features an urban and residential landscape (highlighted in gray). The central area, on the other hand, is predominantly green, with parks, tree-lined spaces, historical buildings, and a river (highlighted in green). This area has fewer points on average due to the lower presence of tall buildings and the LiDAR's limitations in accurately capturing water bodies. The rightmost part presents a more heterogeneous environment, with hilly terrain, vegetation, and large houses (highlighted in orange). The dataset's value lies in its wide variety of landscapes, despite all the data coming from a single city.
Regarding the data structure of Turin3D, the point cloud is stored in the LAS 1.4 format, which provides a structured framework for encoding each point with its attributes. Each point in the dataset contains rich information captured through LiDAR sensors, including the following attributes:

- *(X, Y, Z).* 3D coordinates of the LiDAR point, stored as long integers. The actual coordinates are computed using scale and offset values.
- *Intensity.* The magnitude of the returned laser pulse. Normalized to a 16-bitunsigned integer, with values depending on sensor characteristics.
- *RGB.* Color values associated with the LiDAR point, typically derived from an auxiliary camera. Normalized to 16-bit.
- *Return Number.* Indicates which return this point corresponds to within a single emitted pulse.
- *Number of Returns.* Total number of returns recorded for the given pulse.
- *Scan Direction.* Indicates whether the scan was moving left-to-right or right-to-left.
- *Flight Line Edge Flag.* Indicates whether the point is at the edge of the flight line.
- *Scan Angle.* The angle at which the laser pulse was emitted, relative to nadir.
- *User Data.* Field for custom user-defined information.

- *Point Source ID*. Identifier for the file or flight line from which the point originated.

- *GPS Time*. Time at which the point was recorded, based on GPS timestamping.

Those listed earlier are all fields provided by the point clouds in the dataset; in the course of the experiments covered in the thesis work, only the following fields were used: *(X,Y,Z)*, *RGB* and *intensity*.

## 3.2 Data Acquisition

The data in this dataset were acquired during an aerial flight, at an altitude of approximately $1km$, conducted on January 28-29 2022. Specifically, the acquisition was possible thanks to the **Leica CityMapper-2**, a hybrid aerial sensor that integrates LiDAR technology and optical imaging, providing high spatial accuracy and full urban coverage. Through this acquisition, 69,591,759 points were collected and each square meter possesses an average resolution of about 51.05 points, thus providing a detailed representation of the urban environment.
The LiDAR system operates at a pulse repetition rate of $2MHz$ and employs a *gateless multiple-pulse-in-air (MPiA)* approach, allowing precise distance measurements with an accuracy of about $3cm$.The scanning pattern is oblique. This ensures a uniform distribution of points in the dataset. As a result, the reconstruction of vertical surfaces, such as buildings, is improved. These surfaces are clearly present in the treated case. In addition, the optical system is equipped with two *150 MP nadiral cameras (RGB and NIR)* and four *150 MP* 45° *oblique cameras (RGB)*, which capture high-resolution images so that LiDAR data can be completed. A total of 20,291 images could be collected through these cameras. After these surveys to obtain a dataset that could be used in semantic segmentation, the LiDAR cloud was merged with the optical images, producing a 3D colorized point cloud. This process preserves both the geometric accuracy of the LiDAR and the radiometric consistency of the photogrammetric data, as illustrated in Figure 3.3. In particular, it can be seen that the dataset detection is accurate in detecting all elements in the city such as buildings, vegetation providing a good reference to be used in 3D semantic segmentation.

| Dataset | Year | # points | Classes | RGB | Intensity | Area | Sensor |
|---|---|---|---|---|---|---|---|
| SensatUrban [14] | 2020 | 2847M | 13 | ✓ | ✗ | 7.64 $Km^2$ | UAV Photogrammetry |
| Swiss3D [20] | 2020 | 226M | 5 | ✓ | ✗ | 2.7 $Km^2$ | UAV Photogrammetry |
| Toronto 3D [18] | 2020 | 78M | 8 | ✓ | ✗ | 1 $Km^2$ | MLS |
| Hessigheim [21] | 2021 | 74M | 11 | ✓ | ✓ | 0.19 $Km^2$ | ULS |
| SUM [15] | 2021 | 19M | 6 | ✓ | ✗ | 4 $Km^2$ | Aerial photogrammetry |
| STPLS3D [19] | 2022 | 794M | 18 | ✓ | ✗ | 7.27 $Km^2$ | Aerial Photogrammetry + SAP |
| ECLAIR [16] | 2024 | 582M | 11 | ✓ | ✗ | 10.3 $Km^2$ | ALS |
| FRACTAL [17] | 2024 | 9261M | 7 | ✓ | ✓ | 250 $Km^2$ | ALS |
| Turin3D | 2025 | 69M | 6 | ✓ | ✓ | 1.422 $Km^2$ | ALS |

**Table 3.1:** Comparison of Turin3D dataset with the representative datasets for 3D semantic segmentation in urban scenarios. MLS: Mobile Laser Scanning system, ULS: Unmanned Laser Scanning system, ALS: Airborne Laser Scanning system, SAP: Synthetic Aerial Photogrammetry.

## 3.3 3D Point Cloud Processing

After data collection, a reconstruction step was necessary to obtain more complete and geometrically improved point clouds that will compose the dataset. As shown in Section 3.2 there was an interaction of the LiDAR data with the aerial images. For this purpose, *Agisoft Metashape* and *nFrames SURE*, two advanced 3D photogrammetry and reconstruction software tools, were used in order to obtain point clouds that were as detailed as possible. It could be seen that integration between LiDAR data and oblique imagery can give better results, as it allows better reconstruction of vertical surfaces. Also through this data integration it was possible to acquire very useful additional information such as *intensity*, which is important for point classification. In fact, images make it possible to classify land use and land cover, while LiDAR data make it possible to distinguish soil, vegetation, and buildings, offering a significant advantage due to their sunlight-independent acquisition capability. Therefore, the use of this combination is essential for 3D models as it ensures that the reconstructed point cloud is geometrically accurate and radiometrically enriched.

## 3.4 Semantic labels taxonomy

The definition of semantic labels followed three basic principles:

- Each class must be clearly distinguishable from the others, providing high heterogeneity between classes and high homogeneity within each class.

- The classes selected for the taxonomy of this dataset were extracted from those present in literature datasets to ensure uniformity. Additionally, they were chosen to be relevant for urban planning applications.

- The classification must bring value to subsequent analyses, with a focus on urban planning and green area applications.

The classes to be used, along with special case scenarios such as bridge structures and different types of flooring (e.g., terrace flooring), were discussed and agreed upon among the annotators to ensure a consistent and uniform definition.

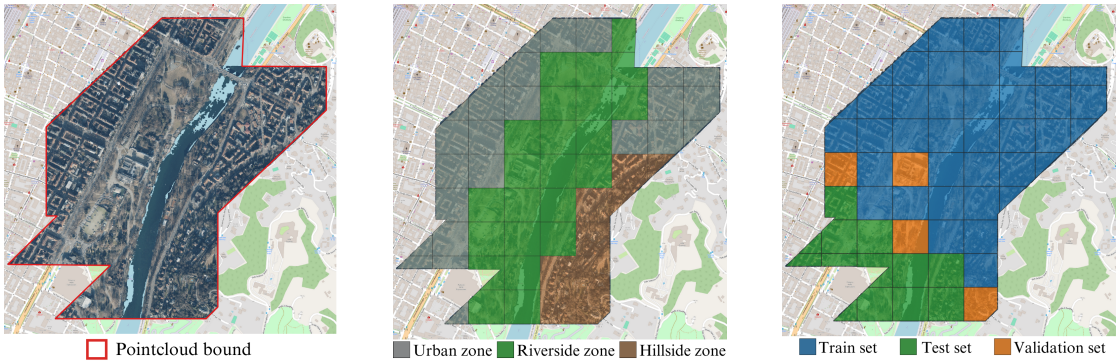
A taxonomy consisting of six distinct semantic labels was adopted to simplify the classification task. The number of classes is lower than in other datasets to avoid labels that are too similar and difficult for human annotators to distinguish. The chosen classes and their identifiers are listed below:

- *Unassigned* (0): is used to include points derived from noise in the acquisition and reconstruction process; these points can be found clustered in masses too small to be classified.
- *Soil* (1): natural surfaces, meadows.
- *Terrain* (2): any other artificial grounds, such as streets or sidewalks.
- *Vegetation* (3): trees, shrubs, bushes, and any other kind of low and high vegetation.
- *Building* (4): walls, fences, barriers, residential and historic buildings.
- *Street elements* (5): cars, trucks, poles, benches.
- *Water* (6): river, water canals, pools.

A key feature of this taxonomy is its ability to distinguish between natural and artificial terrain, an aspect found in some other datasets in the literature but not all.

## 3.5 Data splitting

The dataset was split into approximately 70% for training, 20% for testing, and 10% for validation, it is possible to observe the specific division in Figure 3.2c. However, only the validation and test subsets were manually annotated, while the training set was assigned soft labels, which were used in some experiments conducted as part of this research.




(a) (b) (c)

**Figure 3.2:** The image represents several views of the point cloud contained in the Turin3D dataset, (a) The entire area captured by the point cloud in the city of Turin, shown in RGB. (b) Different areas in the point cloud, categorized by environment. (c) Split for validation, training and test sets

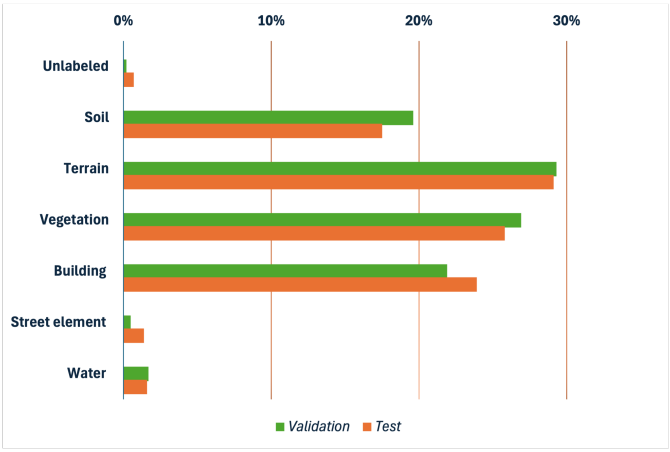



**Figure 3.3:** Distribution of classes in the validation and test set.

When selecting the tiles for each subset, care was taken to ensure that they all contained the same variety of landscapes, preventing any environment from being absent in any split. The figure 3.3 shows how the points for each class are distributed across both the validation and test sets.

## 3.6 Annotation process

As mentioned in Section 3.5, after splitting the dataset, it was necessary to annotate the test and validation subsets. This annotation process consisted of several steps;

the first step involved selecting datasets from the literature that were similar to Turin3D, based on both their taxonomy and the type of sensors used for acquisition. Each class of these datasets was mapped into one of the 6 classes of the proposed taxonomy, a process that will be described in detail in Section 3.7. This concatenated dataset was then used to train the RandlaNet [6] model. By running inference on the Turin3D point clouds, the model generated the initial annotations for the dataset. This part was useful for annotation purposes, but additionally required the manual work by four annotators. After completing the initial annotation of the 17 tiles, a peer review of all the work was done to ensure the consistency of all annotations. The annotation process was supported by comparing the point clouds with satellite images of the city taken in March 2022, and by using additional point features such as *RGB* and *intensity*.

## 3.7 Class mapping across Datasets

In order to ensure a meaningful evaluation of transfer learning, it is crucial to select datasets with taxonomies comparable to that of Turin3D. Based on the literature, the most suitable ones were identified to be used for the experiment. The selected datasets are: **Sensat Urban**[14], **Delft Sum**[15], **ECLAIR**[16], **Fractal**[17], **Toronto3D**[18], **SPTLS3D**[19], **Swiss3D**[20] and **Hessigheim**[21].
The following tables specifically illustrate the taxonomies of each selected dataset and the mapping that had to be done based on the classes in Turin3D.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 0 | Ground | impervious surfaces, grass, terrain | Soil (1) |
| 1 | Vegetation | trees, shrubs, hedges, bushes | Vegetation (3) |
| 2 | Building | commercial / residential buildings | Building (4) |
| 3 | Wall | fence, highway barriers, walls | Building (4) |
| 4 | Bridge | road bridges | Terrain (2) |
| 5 | Parking lots | parking lots | Terrain (2) |
| 6 | Rail | railroad tracks | Terrain (2) |
| 7 | Traffic road | including main streets, highways | Terrain (2) |
| 8 | Street Furniture | benches, poles, lights | Street Element (5) |
| 9 | Car | cars, trucks, HGVs | Street Element (5) |
| 10 | Footpath | walkway, alley | Terrain (2) |
| 11 | Bike | bikes / bicyclists | Street Element (5) |
| 12 | Water | rivers / water canals | Water (6) |

**Table 3.2:** Mapping of the Sensat Urban dataset classes to the Turin3D taxonomy, including their labels and descriptions.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 0 | Unclassified | Incomplete objects. | Unlabeled (0) |
| 1 | Terrain | Roads, bridges, grass fields, and impervious surfaces. | Unlabeled (0) |
| 2 | High Vegetation | Trees, shrubs, and bushes. | Vegetation (3) |
| 3 | Building | Houses, high-rises, monuments, and security booths. | Building (4) |
| 4 | Water | Rivers, sea, and pools. | Water (6) |
| 5 | Vehicle | Cars, buses, and lorries. | Street Element (5) |
| 6 | Boat | Boats, ships, freighters, and sailboats. | Unlabeled (0) |

**Table 3.3:** Mapping of the Delft Sum dataset classes to the Turin3D taxonomy, including their labels and descriptions.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 0 | Undefined | No specific classification assigned. | Unlabeled (0) |
| 1 | Unassigned | A catch-all category for non-subject points. Anything that is not on the class list is classified as Unassigned. | Unlabeled (0) |
| 2 | Ground | All points representing the Earth's surface,including soil, pavement, roads, and the bottom of water bodies. | Unlabeled (0) |
| 3 | Vegetation | All points representing organic plant life, including trees, low shrubs, and tall grass of various heights. | Vegetation (3) |
| 4 | Buildings | Man-made structures with roofs and walls, such as houses, factories, and sheds. | Building (4) |
| 5 | Noise | Points that do not belong to any relevant class. | Unlabeled (0) |
| 6 | Transmission Wires | High-voltage wires for long-distance transmission from power plants to substations, including transmission ground wires. | Street Element (5) |
| 7 | Distribution Wires | Lower-voltage overhead wires distributing electricity from substations to end users, including span guy wires and communication wires. | Street Element (5) |
| 8 | Poles | Utility poles supporting different types of wires or electroliers. Includes poles with either transmission or distribution wires, as well as down guy wires, crossarms, and transformers. | Street Element (5) |
| 9 | Tower (Transmission) | Large structures supporting transmission wires, characterized by steel lattices and cross beams. | Building (4) |
| 10 | Fence | Barriers, railings, or other upright structures, typically of wood or wire, enclosing an area of ground. | Building (4) |
| 11 | Vehicles | All wheeled vehicles that can be driven, including cars, buses, and trucks. | Building (4) |

**Table 3.4:** Mapping of the ECLAIR dataset classes to the Turin3D taxonomy, including their labels and descriptions.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 1 | Unclassified | All points that do not belong in any of the other classes. Includes vehicles, animals, people, temporary objects, wood piles, etc. | Unlabeled (0) |
| 2 | Ground | Points on natural and artificial ground. Bridge decks are excluded. | Unlabeled (0) |
| 3 | Low Vegetation | Trees, shrubs, bushes, ferns, reeds, etc. Ground-level vegetation (<20 cm) is included only if enough ground points are present locally. | Soil (1) |
| 4 | Medium Vegetation | Vegetation between 0.5 m and 1.5 m height. | Vegetation (3) |
| 5 | High Vegetation | Vegetation taller than 1.5 m. | Vegetation (3) |
| 6 | Building | Permanent structures (>10 $m^2$). Includes houses, castles, mills, towers, chimneys, fortifications, roofs, facades, and some lightweight structures. | Building (4) |
| 9 | Water | Points on rivers, lakes, seas, or oceans. | Water (6) |
| 17 | Bridge | Bridge structures. | Terrain (2) |
| 64 | Permanent Structure | Aboveground objects that define the landscape but are not buildings, vegetation, or bridges. Includes wind turbines, antennas, pylons, and bridge elements. | Street Element (5) |
| 65 | Artefact | Points that do not correspond to real-world objects or terrain. | Unlabeled (0) |
| 66 | Synthetic | Artificial points created under bridges and on water surfaces to ensure coherent digital models. | Unlabeled (0) |

**Table 3.5:** Mapping of the Fractal dataset classes to the Turin3D taxonomy, including their labels and descriptions.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 0 | Unclassified | | Unlabeled (0) |
| 1 | Road | Paved road surfaces, including sidewalks, curbs, parking lots. | Terrain (2) |
| 2 | Road Markings | Pavement markings including driving lines, arrows, pedestrian crossings. | Terrain (2) |
| 3 | Natural | Trees, shrubs, not including grass and bare soil. | Vegetation (3) |
| 4 | Building | Any parts of low and multi-story buildings, store fronts. | Building (4) |
| 5 | Utility Line | Power lines, telecommunication lines over the streets. | Street Element (5) |
| 6 | Pole | Utility poles, traffic signs, lamp posts. | Street Element (5) |
| 7 | Car | Moving cars and parked cars on roadsides and parking lots. | Street Element (5) |
| 8 | Fence | Vertical barriers, including wooden fences, walls of construction sites. | Building (4) |

**Table 3.6:** Mapping of the Toronto3D dataset classes to the Turin3D taxonomy, including their labels and descriptions.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 0 | Ground | Grass, paved roads, dirt, sidewalk, parking lots, etc. | Terrain (2) |
| 1 | Building | Commercial, residential, educational buildings. | Building (4) |
| 2 | Low Vegetation | 0.5 m $<$ vegetation height $<$ 2.0 m. | Vegetation (3) |
| 3 | Medium Vegetation | 2.0 m $<$ vegetation height $<$ 5.0 m. | Vegetation (3) |
| 4 | High Vegetation | Vegetation height $>$ 5.0 m. | Vegetation (3) |
| 5 | Vehicle | Sedans and hatchback cars. | Street Element (5) |
| 6 | Truck | Pickup trucks, cement trucks, flat-bed trailers, trailer trucks, etc. | Street Element (5) |
| 7 | Aircraft | Helicopters and airplanes. | Street Element (5) |
| 8 | Military Vehicle | Tanks and Humvees. | Street Element (5) |
| 9 | Bike | Bicycles. | Street Element (5) |
| 10 | Motorcycle | Motorcycles. | Street Element (5) |
| 11 | Light Pole | Light poles and traffic lights. | Street Element (5) |
| 12 | Street Sign | Road signs at the side of roads. | Street Element (5) |
| 13 | Clutter | City furniture, construction equipment, barricades, and other 3D shapes. | Street Element (5) |
| 14 | Fence | Timber, brick, concrete, metal fences. | Building (4) |
| 15 | Road | Asphalt and concrete roads. | Building (4) |
| 17 | Windows | Glass windows. | Building (4) |
| 18 | Dirt | Bare earth. | Soil (1) |
| 19 | Grass | Grass lawn, wild grass, etc. | Soil (1) |

**Table 3.7:** Mapping of the STPLS3D dataset classes to the Turin3D taxonomy, including their labels and descriptions.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 1 | Terrain | Natural terrain (e.g., grass or soil), impervious terrain (e.g., road or sidewalk), and water areas (e.g., river or lake). | Unlabeled (0) |
| 2 | Building | Built structures and man-made constructions. | Building (4) |
| 3 | Urban asset | Traffic light, pole, crane, public transportation stop, trash bin, etc. | Street Element (5) |
| 4 | Vegetation | Tree or bush. | Vegetation (3) |
| 5 | Vehicle | Car, bike, scooter, etc. | Street Element (5) |

**Table 3.8:** Mapping of the Swiss3D dataset classes to the Turin3D taxonomy, including their labels and descriptions.

| Class | Label | Description | Mapping |
|---|---|---|---|
| 0 | Low Vegetation | Grass, small plants, and ground-level vegetation | Soil (1) |
| 1 | Impervious Surface | Paved areas such as roads, sidewalks, and plazas | Terrain (2) |
| 2 | Vehicle | Cars, trucks, and other road vehicles | Street Element (5) |
| 3 | Urban Furniture | Benches, streetlights, poles, and other street installations | Street Element (5) |
| 4 | Roof | Upper surfaces of buildings, including sloped and flat roofs | Building (4) |
| 5 | Facade | Vertical walls of buildings, including windows and doors | Building (4) |
| 6 | Shrub | Bushes and medium-sized vegetation | Vegetation (3) |
| 7 | Tree | Large trees, including trunks and canopies | Vegetation (3) |
| 8 | Soil/Gravel | Unpaved surfaces, such as dirt paths and gravel areas | Soil (1) |
| 9 | Vertical Surface | Vertical non-building elements like retaining walls | Building (4) |
| 10 | Chimney | Smoke stacks and ventilation structures on rooftops | Building (4) |

**Table 3.9:** Mapping of the Hessigheim dataset classes to the Turin3D taxonomy, including their labels and descriptions.

# Chapter 4

# Methodology

This chapter provides a detailed overview of the methodologies used throughout the thesis and why it was necessary to investigate these fields. It covers the neural network architectures used, the datasets selected from the literature, and the specific parameters chosen for each experiment.

## 4.1 Problem Statement

Semantic segmentation of 3D point clouds is an essential task in various domains, including autonomous driving, urban mapping, and territorial planning. However, as mentioned earlier, several challenges arise due to the irregularity of the data and their variability in distribution. Additionally, the availability of annotated data is often limited, as is the case of Turin3D dataset under consideration. For this reason, techniques are often employed to leverage well-annotated datasets and transfer the acquired knowledge to domains with sparser annotations. However, this approach can introduce challenges in model generalization, particularly in urban environments, where city landscapes can vary significantly. In this study, one of the main challenges was identifying datasets with urban landscapes as similar as possible to that of Turin3D.
To address the challenges described above, two main techniques were used: **transfer learning** and **semi-supervised learning with pseudo-labeling**.
The following experiments will refer to the source domain under consideration as $\mathcal{D}_S = \{(x_i^S, y_i^S)\}_{i=1}^{N_S}$ composed of $N_S$ point clouds from literature datasets, where $x_i^S \in \mathbb{R}^{P_i \times F}$ denoted a point cloud with $P_i$ points of $F$ features, and $y_i^S \in \mathcal{C}^{P_i}$ denoted point-wise labels from class set $\mathcal{C} = \{$Unassigned, Soil, Terrain, Vegetation, Building, Street Element, Water$\}$, according to the taxonomy proposed in Section 3.4.The datasets selected from the literature included **Sensat Urban** [14], **DELFT SUM** [15], **Toronto3D** [18], **Fractal** [17], **STPLS3D**[19],

**Swiss3D** [20], **Hessigheim** [21] and **ECLAIR** [16], with each dataset's original classes mapped to exactly one class of $\mathcal{C}$. While the following formulation will be used to refer to *Selected Datasets*, composed of **Sensat Urban**[14], **Delft Sum**[15], **Fractal**[17], **Toronto3D**[18], **SPTLS3D (real)**[19], **Swiss3D**[20] and **Hessigheim**[21],$\mathcal{D}_{Sel} = \{(x_i^{sel}, y_i^{sel})\}_{i=1}^{N_{sel}}$. The target domain, **Turin3D**, is rappresented by $\mathcal{D}_T = \{(x_j^T)\}_{j=1}^{N_T^{train}} \cup \{(x_k^T, y_k^T)\}_{k=1}^{N_T^{val}} \cup \{(x_l^T, y_l^T)\}_{l=1}^{N_T^{test}}$ with unlabeled training and labeled validation and test sets, consisting of $N_T^{train}$, $N_T^{val}$ and $N_T^{test}$ point clouds, respectively. Four different types of architectures were used for the transfer learning experiments so that it could be evaluated which one obtained the best results to be used in the subsequent experiments. These neural network architectures are $\mathcal{A}$ ={RandLANet[6], PointTransformer[25], SparseConv[13], used in the experimental approaches described in the following sections. All the methodologies used in the course of the work are presented in detail below.

## 4.2 Transfer learning

The technique of **transfer learning**[22] allows knowledge acquired in one domain to be applied to a different but related domain. This approach is particularly useful in scenarios where there is limited labeled data available in the target domain, but abundant labeled data exists in the related source domain.
In this research, the transfer learning technique was employed to address the issue of limited annotated data in the Turin3D dataset. For Turin3D the annotation process was carried out only on the test and validation subsets, since annotation can be a complicated and costly process. Therefore, it was necessary to explore methodologies that could be applied despite the lack of annotations in the training subset. To this end, the source domain for the transfer learning experiments consisted of two dataset variations: one including all selected datasets from the literature ($\mathcal{D}_S$) and another with only a subset of them ($\mathcal{D}_{Sel}$). These datasets were introduced in Section 2.5, while the necessary mapping to adapt them to Turin3D was illustrated in Section 3.4. This remapping enables the integration of case reports from different datasets into a unified representation, facilitating the development of a robust segmentation model.
In the first case, the focus was on concatenating all the datasets selected from the literature. These datasets were chosen based on the mode of acquisition and their representation of urban environments, similar to those covered in the current work. After this first step in which all datasets were used, it has been evaluated if excluding some of them would yield better results. Thus, given that ECLAIR[16] and STPL3D(Synth)[19] were the most different datasets from Turin3D in terms of taxonomy, they were excluded. ECLAIR[16] is particulary focused on the semantic classification of electrical elements, for example pole or power transmission. These

elements are not present in the Turin3D dataset and they are far from the purpose of its classification. In the case of STPLS[19], the synthetic component has been excluded. Because combining real and *synthetic data*[26] in semantic segmentation is a challenge since they are very different. Synthetic point clouds are generated using software such as LiDAR simulations or 3D models, which makes them more "perfect" than real data. Real data, on the other hand, are noisy and affected by factors such as lighting, occlusions, and environmental variability. These differences can make it difficult to apply models trained on synthetic data to real data. In particular, synthetic data often exhibit substantial differences in color compared to real data, which could lead to a deterioration in the model's performance.
Another issue considered was whether to include or exclude the Sensat Urban dataset from the training phase. Because, the dataset creators did not release labels for the test set, making it impossible to perform tests when merging it with the other datasets.
Therefore these different source domains were used for training and each architecture $a \in \mathcal{A}$ was used to obtain:

$$\theta_a^* = \arg\min_{\theta} \mathcal{L}(f_\theta^a, \mathcal{D}_S) \tag{4.1}$$

These models were then evaluated on both on the datasets used for training ($\mathcal{D}_S^{\text{test}}$ or $\mathcal{D}_{Sel}^{\text{test}}$), based on the experiment, and on Turin3D test set ($\mathcal{D}_T^{\text{test}}$). This approach enabled a comprehensive evaluation of the models' performance across different urban contexts.

## 4.3 Semi-Supervised Learning

In the context of 3D semantic segmentation of point clouds, as discussed in Section 2.6, most common scenarios include partially labelled data, as manual annotation is both expensive and time-consuming. To address this challenge, semi-supervised learning techniques can be employed to leverage partially annotated or pseudo-labeled data. Based on the transfer learning results, the best-performing architecture on the Turin3D validation set was selected:

$$a^* = \arg\max_{a \in \mathcal{A}} \text{mIoU}(f_{\theta_a^*}, \mathcal{D}_T^{\text{val}}) \tag{4.2}$$

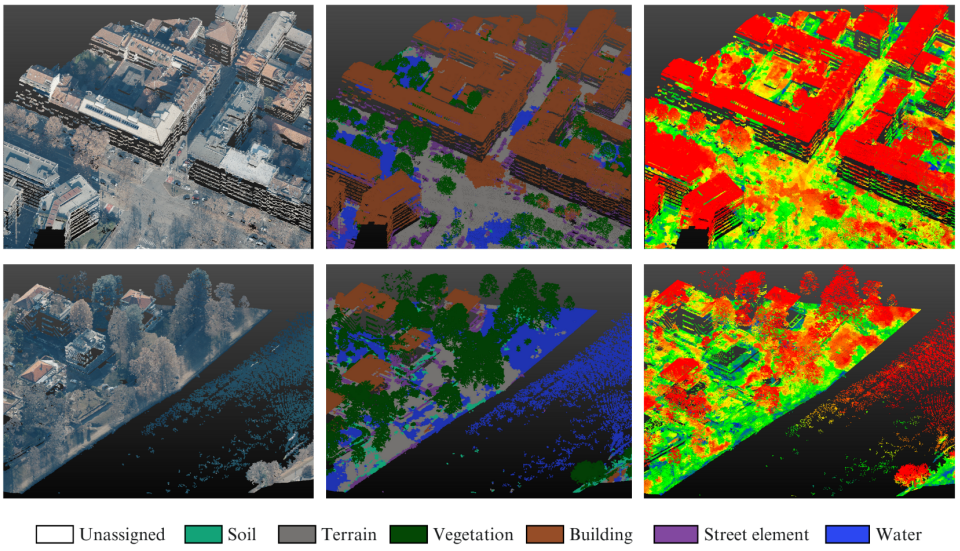



**Figure 4.1:** The image shows an example from the Turin3D training subset: on the left, the dataset in natural colors (RGB), in the middle, the pseudo-labels following the dataset's semantic classes, and on the right, the confidence scores for each point. These labels are pseudo-labels generated by the best-performing transfer learning model, not manually annotated.

The selected model was used to generate predictions on the unlabeled Turin3D training set $\mathcal{D}_T^{train}$. Each point was assigned the class label with the highest confidence score, but only if that score exceeded a class-specific confidence threshold $\tau_c$, minimizing the risk of propagating errors due to model uncertainty. This filtering resulted in a set of high-confidence pseudo-labels $\hat{y}_j^T = f_{\theta_{a^*}^*}(x_j^T)$ for a subset of points in the training set. In fact, the pseudo-labelled point cloud had less points than the original one due to this filtering step.
Across all experiments, the generation of pseudo-labels followed a specific methodology. Point cloud labels were filtered to retain only those that met a predefined confidence threshold. In particular, the threshold for each class was determined as a weighted average of the confidence scores obtained from the model that performed best in the transfer learning experiments. Specifically, the formula used to calculate the threshold is as follows:

$$\tau_c = \sum_{u \in \mathcal{U}_c} u \cdot \frac{n_{u,c}}{\sum_{v \in \mathcal{U}_c} n_{v,c}} \tag{4.3}$$

where $\tau_c$ represents the confidence threshold for class $c$, $\mathcal{U}_c$ is the set of unique confidence values observed for points predicted as class $c$, $u$ is a specific unique confidence value, $n_{u,c}$ is the count of points predicted as class $c$ with confidence

value $u$, and $\sum_{v \in \mathcal{U}_c} n_{v,c}$ is the total number of points predicted as class $c$ (ensuring that the weights sum to 1).
In this phase, the intensity was incorporated as additional point's feature, so the features used become $F = \{x, y, z, R, G, B, \text{Intensity}\}$. The latter could provide valuable information to further refine the training process and improve model robustness.
In the following sections it will be detailed the various methodologies used in semi-supervised learning and the different experiments done to improve the model and select pseudo-labels that are most accurate as possible.

### 4.3.1 Uniform Confidence Threshold

As a first experiment, a fixed confidence threshold of 0.85 was used for all semantic classes. This value was based on the confidence distribution across all classes, assuming it would include enough pseudo-labels despite distribution variations. A high confidence threshold reduces the number of pseudo-labels but increases their reliability. However some classes may remain underrepresented at this threshold. Conversely, the selection of a lower threshold could increase the risk of propagating model errors.

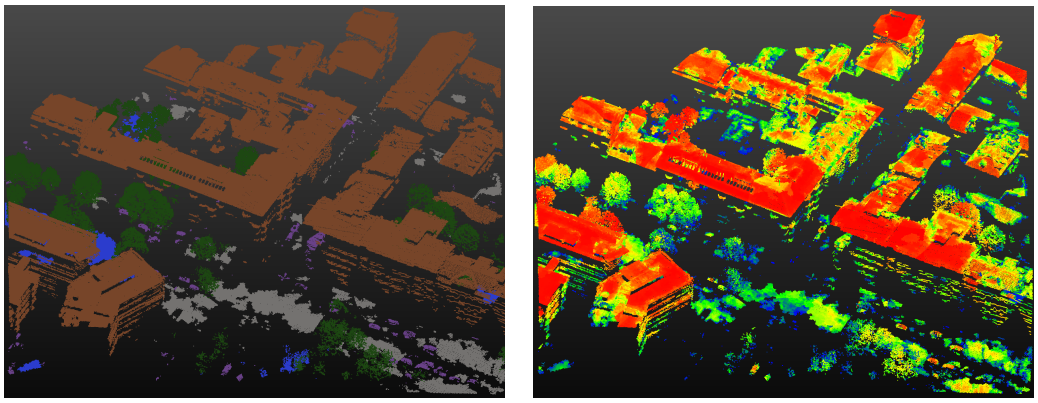


**Figure 4.2:** It is possible to observe selected Pseudo-labels with a confidence threshold of 0.85. The pseudo-labels can be observed on the left, and the confidence representation on the right.

An example of the pseudo-labels considered when using a fixed threshold is shown in Figure 4.2. As mentioned earlier, this illustrates that some classes are underrepresented due to their lower confidence values. This approach was initially chosen for its simplicity. However, after evaluating the results, it was decided to use **class-specific confidence thresholds** in subsequent experiments. As shown in Figure 4.2, some classes, such as soil and street elements, almost disappear completely when the threshold is set to 0.85. Since each class has a very different

confidence distribution, choosing a single threshold turns out not to be the most effective approach.

### 4.3.2 Class-Specific Confidence Threshold

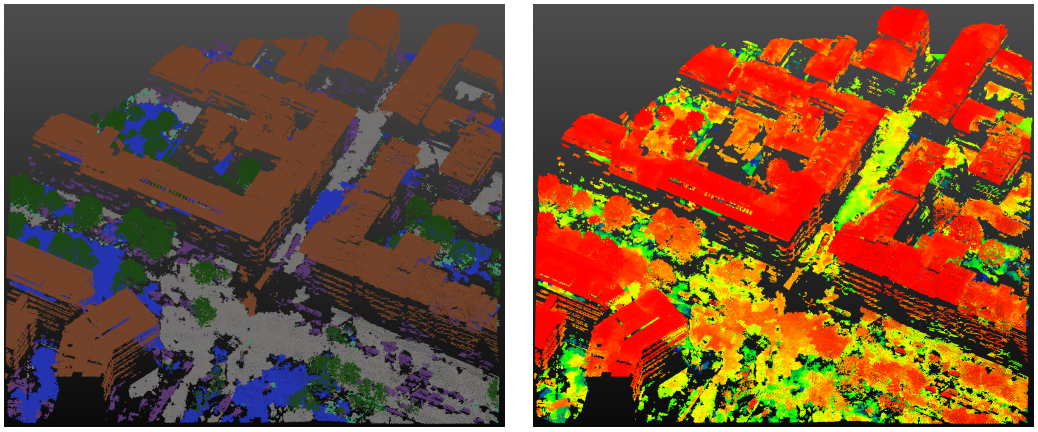


**Figure 4.3:** It is possible to observe selected Pseudo-labels with Class-Specific Confidence Threshold. The pseudo-labels can be observed on the left, and the confidence representation on the right.

As mentioned in the previous section, using the same threshold for all classes is not an optimal solution. As shown in Figure 4.4, the confidence distribution in the Turin3D training subset varies significantly across different semantic classes. Furthermore, using this approach, some classes could be underrepresented. For example, the soil class (1) has a lower confidence distribution, making it more likely to be excluded, while the water class (6) consistently shows high confidence values. To address this, a different confidence threshold is used for each class. Specifically, the threshold for each class is determined by the weighted average of the confidence scores, as represented by the dotted line in Figure 4.4. However, additional considerations were necessary to balance classes that appeared unbalanced. For all classes, the threshold was set based on the weighted average, except for two classes: soil (1) and water (6). As shown in Figure 4.1, the model frequently tends to confuse points belonging to class 1 with those belonging to class 6. To mitigate this problem, a lower threshold was chosen for the *soil class* and a higher threshold for the *water class*. This modification allowed more points to be included in the former and limited the number of points in the latter, thus helping to balance the pseudo-labeling. In conclusion, the thresholds selected for each class are: *Soil (1): 0.1*, *Terrain (2): 0.6*, *Vegetation (3): 0.8*, *Building (4): 0.8*, *Street Element (5): 0.5* and *Water (6): 0.9*.

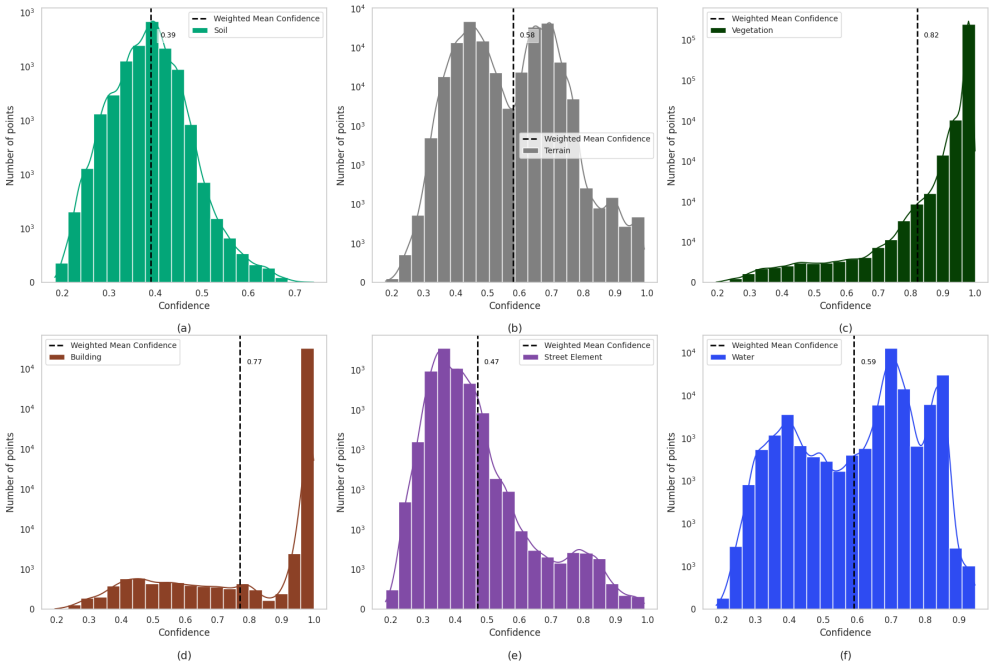



**Figure 4.4:** Confidence distributions in the training subset obtained from the inference of the best-performing model selected via transfer learning.

In conclusion custom thresholds for each class can lead to a better balance of pseudo-labels than seen previously with a fixed threshold. However other solutions that could improve the model will be explored in the following sections.

### 4.3.3 Iterative Pseudo-Label Refinement

One of the main limitations of using pseudo-labels is the potential introduction of noise and uncertainty due to the absence of actual labels. To address this issue, iterative strategies could be used to refine the pseudo-labels, gradually updating them to enhance the reliability of the labels during training. As mentioned in Section 4.3.1, the selection of appropriate confidence thresholds is essential for determining pseudo-labels. Therefore, the first step in this experiment was to filter the pseudo-labels based on the previously defined thresholds; in this way it is more probable to keep only the most reliable ones.
For this stage of the experiments, the chosen configuration consisted of 3 iterations, each running for 200 epochs. This setup was selected as it provided a balance between observing potential improvements and avoiding excessive resource consumption. At each iteration, a re-training of the model was performed using new pseudo-labels. These new pseudo-labels were obtained by running inference on the training subset of Turin3D using the model from the previous iteration. The best

checkpoint, selected based on the highest mIoU achieved on the validation subset, was used for inference. This iterative cycle allows the quality of the labels to be progressively refined, taking advantage of the self-correction of the model as it progresses through each iteration. The goal is to reduce noise in the pseudo-labels and improve the generalization of the model.
However, using the same confidence thresholds at each iteration to select pseudo-labels may lead them to be no longer aligned with the updated confidence scores from subsequent inferences. As a result, the expected improvements may not be achieved.

### 4.3.4 Dynamic Confidence Thresholding in Iterative Training

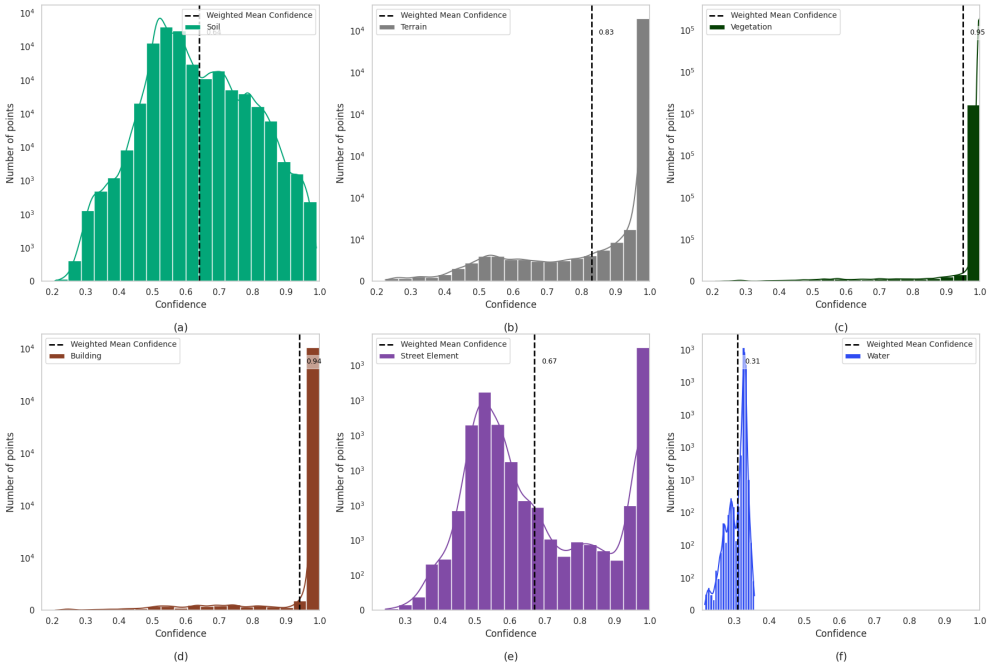



**Figure 4.5:** Confidence distributions in the training subset obtained from inference of the trained model after the first iteration.

In the iterative training process, confidence thresholds play a crucial role in selecting pseudo-labels and ultimately determining model performance. Using a fixed threshold for every iteration may not be optimal, as confidence scores evolve over time. In fact, each iteration generates a new training subset based on the model's inference from the previous iteration. The difference in confidence distributions can be observed between the initial training subset, shown in Figure 4.4, and the one

obtained after inference following the first training iteration, presented in Figure 4.5. **Dynamic confidence thresholds** aim to address this issue by adjusting threshold values at each iteration based on the updated predictions, ensuring a more adaptive and effective selection of pseudo-labels. The iterative process followed these steps:

1. **Initial confidence filtering**. Pseudo-labels were selected based on the thresholds defined for each class discussed above.

2. **Inference on Training Set**. After selecting the pseudo-labels, training was performed for each iteration. Following each iteration, inference was conducted on the Turin3D training subset to obtain the new pseudo-labels. This procedure closely followed the approach described in the previous experiment.

3. **Adjustment of dynamic confidence threshold**. This is the main difference from the previous experiments. At this stage, weighted averages of the total number of points with specific confidence for each class are calculated. This allows for a representation of how the points are distributed in the new training subset obtained from inference.

This process was repeated iteratively for 3 cycles, each consisting of 200 epochs, as described earlier. This allowed for a gradual improvement in the quality of the pseudo-labels and, consequently, the performance of the model.

This iterative process was also conducted using confidence to weight the loss function. By incorporating confidence scores, more importance was given to errors where the model was more confident, i.e., the pseudo-labels were more accurate (higher confidence). On the other hand, low confidence predictions had a reduced impact on the optimization process, as the pseudo-labels used may have been inaccurate. This approach aims to refine the training process by leveraging trust not only to select pseudo-labels for model training, but also to correct them.
Mathematically, the loss function was modified as follows:

$$L = \frac{1}{N} \sum_{i=1}^{N} w_{y_i} \cdot c_i \cdot \mathcal{L}_{\text{CE}}(f(x_i), y_i) \tag{4.4}$$

where $N$ is the total number of samples, $f(x_i)$ is logit for $x_i$ input, $y_i$ is ground truth of sample $x_i$, $\mathcal{L}_{\text{CE}}$ is **Cross-Entropy Loss**, $c_i$ is the confidence for input $x_i$ and $w_i$ is the weight of the class $y_i$ determined by the distribution of classes in the dataset.

## 4.4 Experimental setup

### 4.4.1 Data Augmentation

3D point clouds of urban environments are significantly influenced by a variety of environmental factors. Data collected in cities can be affected by variations in illumination due to atmospheric conditions, different capture angles, or the movement of subjects within the scene. Additionally, the sensors used to acquire the data may introduce inaccuracies. In the context of this research, LiDAR sensors as explained in Section 2.1 may produce disturbances in the collected data due to reflective surfaces.
**Data augmentation**[27] is a technique used to improve model performance, particularly when the available datasets are limited or unrepresentative. It allows increasing the volume, quality and diversity of training data by applying transformations on existing data. In fact, using it helps models generalize better, reducing the risk of overfitting and making them more robust to variations in the input data, such as illumination, rotations or geometric distortions. During the experiments, data augmentation techniques were applied to improve the performance of the models and enhance their generalization ability to adapt to the new domain. The following data augmentation techniques were used:

- **Normalize**. In this technique, coordinates are scaled in the same range so as to ensure a consistent distribution with the least possible dependence on different scales of the input data. In this case, linear normalization is applied, which is defined as:

$$\mathcal{D}'_S = \frac{\mathcal{D}_S - \mu(\mathcal{D}_S)}{s} \tag{4.5}$$

  where $\mu(\mathcal{D}_S)$ represents the mean of the points that compose the source domain $\mathcal{D}_S$ , while $s = max(max\mathcal{D}_S - min\mathcal{D}_S)$.

- **Recentering**. This technique has the purpose of recentering the coordinates along the axes specified as parameters in this case all axes were chosen, thus reducing the sensibility to the absolute position of the points favoring a homogeneous representation. The recentering augmentation can be expressed as:

$$\mathcal{D}_S^{(d)} = \mathcal{D}_S^{(d)} - \mu(D_s)^{(d)} \tag{4.6}$$

  where $\mathcal{D}_S^{(d)}$ represents the data along the $d - th$ dimension, while $\mu(D_s)^{(d)}$ represents the mean of the points that compose the source domain $\mathcal{D}_S$ along d-th dimension; $d \in \{0,1,2\}$ indicates the possible dimensions along which the centering operation occurs.

- **Rotate**. In this technique, careful rotation is applied to multiple axes; in the case discussed, the *"all"* parameter was used to allow rotation along all directions. The maximum rotation angle was set to 30°, as this augmentation was particularly useful for improving performance in the experiments conducted on the Turin3D dataset. Given that the dataset includes tiles representing hilly areas with multiple elevation layers rather than a flat terrain, applying such rotations helped the model generalize better to variations in elevation and terrain structure. The rotation transformation can be expressed as:

$$\mathcal{D}'_S = \{R\mathbf{p} \mid \mathbf{p} \in \mathcal{D}_S\} \tag{4.7}$$

$$R = I + \sin\alpha[\mathbf{u}]_\times + (1-\cos\alpha)[\mathbf{u}]^2_\times \alpha \sim U(0, \pi/6) \tag{4.8}$$

where $\mathbf{u} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi \end{bmatrix}, \quad \theta \sim U(0, 2\pi), \quad \phi \sim U\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$

- **RandomHorizontalFlip**. This technique applies a reflection transformation along the specified axes. The formula to be applied is:

$$\mathcal{D}'_S = max(\mathcal{D}_S)^d - \mathcal{D}^d_S, \quad d \in [0,1,2] \tag{4.9}$$

where $\mathcal{D}^d_S$ is for all the data belonging to the source domain for each different value of the axis. In the experiments conducted, the transformation was applied to the x and y axes (so with parameters *[0,1]*), while the z axis was excluded. Because in the case of urban data, the flip along the z axis would be unrealistic, since the ground should always remain at the bottom. The goal of this augmentation is that the model becomes more robust to variations in city and street orientation, improving its generalization ability.

- **ChromaticAutoContrast**. Using an application probability of *0.1*. This technique adjusts color contrast; it works by improving the difference between lighter and darker points, actually increasing contrast. This helps simulate the variations in lighting conditions that occur in real-world scenarios. The transformation can be expressed as:

$$\mathbf{c}_{\min} = \min_i \mathbf{c}_i, \quad \mathbf{c}_{\max} = \max_i \mathbf{c}_i, \quad \mathbf{c}_i = (r_i, g_i, b_i)$$

$$\mathbf{s} = \frac{255}{\mathbf{c}_{\max} - \mathbf{c}_{\min}}, \quad \mathbf{c}'_i = (\mathbf{c}_i - \mathbf{c}_{\min}) \cdot \mathbf{s} \tag{4.10}$$

$$\mathbf{c}^{\text{new}}_i = (1-\lambda)\mathbf{c}_i + \lambda\mathbf{c}'_i, \quad \lambda \sim U(0,1) \tag{4.11}$$

where $\mathbf{c}_i$ represents the *RGB* fields of all points belonging to the domain $\mathcal{D}_S$.

- **ChromaticJitter**. This technique introduces jitter into the color values of the points in the point cloud. It does this by adding small variations to the RGB values of the points, which are usually controlled by a standard deviation value in this case of *0.008*. This technique also aims to make the model less affected by lighting conditions. The function applied to the The function applied to the domain $\mathcal{D}_S$ is defined as:

$$\mathbf{n} \sim \mathcal{N}(0,1)^{N\times 3}, \quad \mathbf{n}' = \mathbf{n} \cdot \sigma \cdot 255 \tag{4.12}$$

$$\mathbf{c}_i^{\text{new}} = \text{clip}\left(\mathbf{c}_i + \mathbf{n}', 0, 255\right) \tag{4.13}$$

  where $\mathbf{n}$ represents a standard Gaussian noise, while also in this case $\mathbf{c}_i$ represents the *RGB* fields of all points belonging to the domain $\mathcal{D}_S$. The *clip* function is used to make sure that the new results remain in the range [0,255].

- **HueSaturationTranslation**. This technique alters the *hue* i.e., the actual color and *saturation* of the dots in the point cloud. In this way, variations can be achieved that can improve the model, making it more robust. In this specific case, *0.3* as the **maximum hue**($h_{max}$) and *0.1* as the **maximum saturation**($s_{max}$) were used as values. The applied function can be expressed as:

$$H' = \text{mod}(H + \Delta(H),1) \tag{4.14}$$

$$\Delta H = (r_{\text{hue}} - 0.5) \cdot 2 \cdot h_{max}, \quad r_{hue} \sim U(0,1)$$

$$S' = clip(S \cdot (1 + \Delta S),0,1) \tag{4.15}$$

$$\Delta S = (r_{saturation} - 0.5) \cdot 2 \cdot s_{max}, \quad r_{saturation} \sim U(0,1)$$

  where $\Delta H$ is a random variation to the hue added to $H$, the same operation is performed at saturation by $\Delta S$.

Table 4.1 shows a schematic summary of the data augmentations applied to the datasets and their corresponding parameters. The configuration that gave the best results was selected for each experiment, ensuring that no deterioration would occur due to aggressive augmentation.

### 4.4.2 Architectures' hyperparameters

During the transfer learning experiments, several neural network architectures were selected from the literature based on their suitability for 3D semantic segmentation. For each architecture, parameters were chosen to achieve a balance between good performance and efficient memory usage. Below is a list of all the networks used and the most relevant parameters:

| Augmentation | Parameters |
|---|---|
| Recenter | dim: [0,1,2] |
| Normalize | methods: Linear |
| RandomHorizontalFlip | axes: [0,1] |
| Rotate | method: All |
| ChromaticAutoContrast | prob: 0.1 |
| ChromaticJitter | std: 0.008 |
| HueSaturationTranslation | hue max: 0.3<br>saturation max: 0.1 |

**Table 4.1:** Overview of augmentations applied to point clouds, including scaling, rotation, translation, and jittering, along with their respective parameter values.

- **RandLA-Net**[6]. The strength of this neural network architecture is that it efficiently processes large-scale data using a combination of *random sampling* and *local feature aggregation*, while keeping memory consumption limited compared with others.
  During the research, the parameters used for the network were as follows:
  - **Number of neighbors:** 16, each point considers the 16 closest neighbors for feature aggregation, capturing local geometric structures.
  - **Number of layers:** 5, it is composed of five hierarchical layers, which progressively extract high-level features by preserving spatial relationships.
  - **Subsampling ratio:** $[4, 4, 4, 4, 2]$, this parameter defines the subsampling factor to be applied at each local aggregation level of the network. This is used to reduce the number of points processed at each level.
  - **Feature size:** $[8, 16, 64, 128, 256, 512]$, represents how the number of feature channels increases through each layer, allowing the network to learn progressively richer information.
  - **Grid size for downsampling:** 0.2, during the downsampling step, the network aims to reduce the point density of the point cloud. In order to enhance computational efficiency, minimizing the number of points to be processed while preserving the geometric representation.
- **Point Transformer V1**[11]. It exploits a self-attention mechanism to effectively capture complex spatial relationships within unstructured 3D data, which is why it was selected to perform the experiments. The parameters chosen are:

  - **Number of transform blocks**[28]:$[2, 3, 4, 6, 3]$, defines the number of transformer blocks at different depths of the network, providing progressive feature refinement.
  - **Voxel size:**0.2, which determines the resolution of voxelization, i.e., the size of each voxel into which the point cloud is divided. A larger value results in more points being clustered within a single voxel. However, if the value is too large, it may lead to a loss of finer details.

- **SparseConv**[13]. It is specifically designed to handle irregular data in point clouds in a way that reduces computational costs and maintains high representation power. In fact, unlike standard convolutional networks that assume dense representations of data, SparseConv operates directly on sparse inputs. The key hyperparameters used during the experiments were:
  - **Multiplier:** 32, it controls the expansion of feature maps across layers, allowing the network to increase its capacity for hierarchical feature extraction.
  - **Voxel size:** 0.2, it determines the resolution of voxelization, again depending on the value of it varies the level of detail.
  - **Residual blocks**: *True*, it enables residual connections within convolutional layers, facilitating gradient flow and improving the training of deeper networks.
  - **Convolution block repetitions:** 1, this specifies how many times each convolutional block is repeated before moving to the next stage, impacting feature refinement.

The parameters were not chosen through grid searches or extensive experiments, as that would have been too time-consuming. Instead, they were selected based on values suggested in the literature for different architectures and recommendations from experiments on various datasets.
All these neural network architectures were used to conduct these experiments and find the model that had the best performance to then be used in the next steps.

# Chapter 5

# Experimental Results

This chapter initially presents the metrics chosen to evaluate the experiments conducted and then provides an analysis of the results obtained. The experiments aim to assess the performance of various approaches, starting with the transfer learning phase followed by the semi-supervised learning approach. The results are discussed in detail, highlighting the effectiveness of each method and its impact on the segmentation quality of the Turin3D dataset. The following section will explore the evaluation process, and a comparison of the models' performance using the selected metrics.

## 5.1 Parameters and configurations

Considering all the experiments, the following fixed parameters and technologies were used. Models were trained using NVIDIA A100 GPUs with Multi-Instance GPU (MIG) partitioning, specifically utilizing 20GB and 40GB MIG slices. Training ran for a total of 200 epochs per session, the batch size was set to 4, with a maximum of 65,536 points per batch element, ensuring a balance between accuracy and memory management. In addition, for each of the 200 epochs, 100 point clouds from the training set and 10 point clouds from the validation set are selected in order to perform training and validation. These selected point clouds correspond to spherical neighborhoods extracted from the original point clouds, with the center of each neighborhood randomly sampled. Finally, an initial learning rate of 0.001 was adopted for model optimization, promoting stable convergence throughout the training process.

## 5.2 Metrics

During this research, quantitative metrics were used to be able to evaluate the performance of the applied models. The metrics selected were **F1 Score** and **Intersection over Union (IoU)**.

- **F1 Score**. It is the harmonic mean of precision and recall; it combines both into a single value to provide a balanced measure of classification quality. The F1 score formula is:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{5.2}$$

  where **TP** is *True Positive*, **FP** is *False Positive*, and **FN** is *False Negative*.

- **Intersection over Union (IoU)**. It is a measure of overlap between the predicted segmentation and the ground truth. It is defined as the ratio of the intersection of the predicted and true regions to their union. The formula for IoU is:

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{5.3}$$

  where $A$ is the predicted region and $B$ is the true region. While **TP** is *True Positive*, **FP** is *False Positive*, and **FN** is *False Negative*.

## 5.3 Transfer Learning

To evaluate the actual benefits of applying transfer learning to semantic point cloud segmentation, a series of experiments were conducted exploiting the architectures discussed in Section 2.4. Table 5.1 presents the performance of selected state-of-the-art networks that were identified as the most promising when trained on a concatenation of existing datasets in the literature. The evaluation was performed on both the concatenated datasets ($\mathcal{D}_S$ and $\mathcal{D}_{Sel}$ ) and Turin3D to identify the best performing model. In addition, all experiments were conducted with and without data augmentation to determine whether this could further improve performance.

| Model | Augmentation | Turin3D | | | | | | | | Test | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | *Soil* | *Terrain* | *Vegetation* | *Building* | *Street Elements* | *Water* | **mIoU** | **F1** | **mIoU** | **F1** |
| RandlaNet | ✗ | 0.0 | 57.31 | 4.12 | 38.83 | 20.52 | 64.03 | 30.8 | 41.48 | 57.2 | 74.67 |
| | ✓ | 12.94 | 41.89 | 60.58 | 63.19 | 28.53 | 0.3 | **34.57** | **49.51** | **59** | **71.94** |
| Point Transformer | ✗ | 0.0 | 1.79 | 21.96 | 6.84 | 0.0 | 0.0 | 5.14 | 8.81 | 12.63 | 16.87 |
| | ✓ | 0.0 | 2.98 | 27.65 | 11.34 | 0.0 | 0.0 | 7.05 | 11.69 | 12.83 | 16.52 |
| SparseConv | ✗ | 0.0 | 0.0 | 0.0 | 17.71 | 0.0 | 0.0 | 2.95 | 15.05 | 17.05 | 30.31 |
| | ✓ | 0.0 | 4.18 | 9.04 | 15.52 | 0.0 | 0.0 | 4.9 | 13.18 | 20.22 | 28.29 |

**Table 5.1:** Results for Transfer learning experiments, with and without augmentations, evaluated on both test sets of literature selected datasets $\mathcal{D}_S$ and labeled test set of Turin3D, considering *mIou* and *F1 score*. For Turin3D also *IoU* per class is reported.

These results highlight how the selected architectures generalize differently on Turin3D. **RandLANet** demonstrates the best performance, achieving 34.57 *mIoU* and an *F1-score* of 49.51 on the dataset under consideration when using augmentation. However, a decrease in performance is observed when testing on datasets from the literature, as well as on those used for training the model (59 *mIoU*). This suggests that the model loses some of its effectiveness when transitioning from the source domain (used for training) to the target domain (used for testing), resulting in lower performance than initially expected. This result highlights a key challenge commonly encountered when transferring knowledge from one domain to another. In the context of urban environments, it emphasizes the difficulty of generalizing across different cities, as each city may present unique environmental elements and distinct conditions. What makes the difference is the use of data augmentation, this leads as can be seen from the table an improvement. In fact for **RandLANet**, a significant enhancement can be observed for the classification of vegetation (+56.58), buildings (+24.36) and soil (+12.94). However, although the performance is better, there may be a reduction in performance with regard to water classification (−64.03). In fact when using this technique there may be improvements for some classes but negatively affect others.
As for other architectures, it is clear that they struggle to generalize, causing a greater decrease of performance than **RandLANet** when switching from one domain to another.
**Point Transformer** achieves in the case of using augmentation only 7.05 of *mIoU* on Turin3D, which seems to be better than the result obtained without augmentation (5.14). While in the case of **SparseConv** a minimal improvement can be seen with augmentation, achieving 4.9 *mIoU*. These networks perform worst when classifying *Soil (1)*, *Street Element (5)*, and *Water (6)*. They often fail to classify these correctly. Without augmentation, the performance worsens even more, with additional classes being misclassified. A possible reason may also be the scarcity or total absence of these classes in some of the datasets examined. This issue may highlight the variability between different cities as well as the differences in

annotation conventions across datasets. For example, the street element class may include several urban elements that vary considerably from one another. These considerations lead to the need for effective domain adaptation techniques in order to overcome the problems mentioned above. In contrast, a common trait among almost all of them is decent performance in recognizing vegetation; this seems to be the most easily transferable semantic class due to its similar shape across different urban environments. Similarly, buildings also transfer relatively well, while road elements and water face more difficulty in transferring between domains due to their higher variability across different urban settings. In conclusion, given the superior performance of **RandLANet with augmentation**, the latter is chosen as the best architecture to continue with transfer-learning and semi-supervised experiments.

| | **Turin3D** | | | | | | | | **Test** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ***Soil*** | ***Terrain*** | ***Vegetation*** | ***Building*** | ***Street Elements*** | ***Water*** | **mIoU** | **F1** | **mIoU** | **F1** |
| Selected with **Sensat Urban**[14] | 10.05 | 43.75 | **81.42** | **72.36** | 16.70 | 8.12 | **38.73** | **49.42** | **78.59** | **67.39** |
| Selected without **Sensat Urban**[14] | 31.99 | 34.19 | 72.13 | 57.09 | 8.16 | 0.0 | 34 | 45.3 | 14.9 | 24.13 |

**Table 5.2:** Transfer learning results using RandlaNet with/without Sensat Urban in training selection. The experiments were evaluated on both test sets of selected datasets $\mathcal{D}_{Sel}$ and labeled test set of Turin3D, considering *mIou* and *F1 score*. For Turin3D also $IoU$ per class is reported.

Alongside these experiments, a focus was placed on the possibility of using only a selection of these, excluding ECLAIR and STPLS3D synth ($\mathcal{D}_{Sel}$). The reason for their exclusion is explained in Section 4.2. The Table 5.2 presents the performance of the model under different configurations. In these experiments, it was also evaluated the impact of adding SensatUrban to the source domain collection of datasets during training. Notably, the inclusion of Sensat Urban generally improves overall performance. Indeed, performance increase significantly for both Turin3D(38.73 $mIoU$ and 49.42 *F1*) and source domain test set(78.59 $mIoU$ and 67.39 *F1*). For the selected datasets in particular, the difference is especially pronounced, with an increase of +63.69 in $mIoU$ when Sensat Urban is used. This suggests that Sensat Urban provides valuable features that significantly enhance generalization. Regarding $IoU$ by class, most categories show an increase. Furthermore, the Water class improves substantially, reaching 8.12 $IoU$, whereas it disappears completely without Sensat Urban. The only class that experiences a decline is Soil, where $IoU$ drops from 31.99 to 10.05.

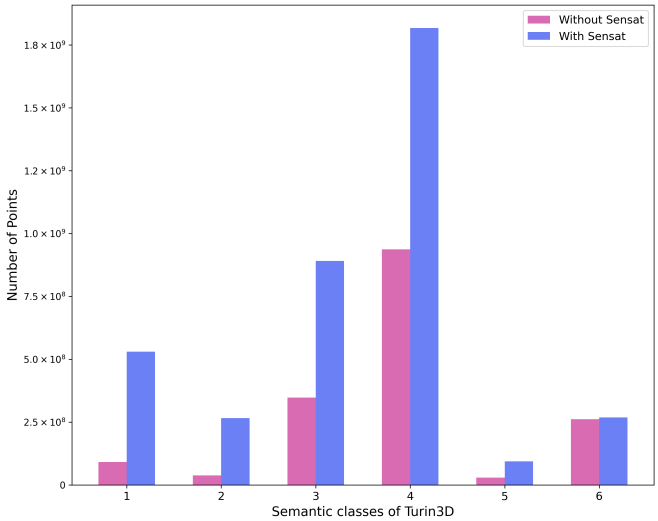



**Figure 5.1:** Number of points per class in the training dataset, comparing the inclusion and exclusion of Sensat in the datasets used for transfer learning experiments.

In conclusion, Sensat Urban significantly improves the generalization of the model, although some classes benefit more than others. This may depend largely on their distribution within the source domain, as shown in Figure 5.1. For example, for the Vegetation and Building classes, Sensat Urban increases the presence of points belonging to these classes, leading to improved model performance for them.

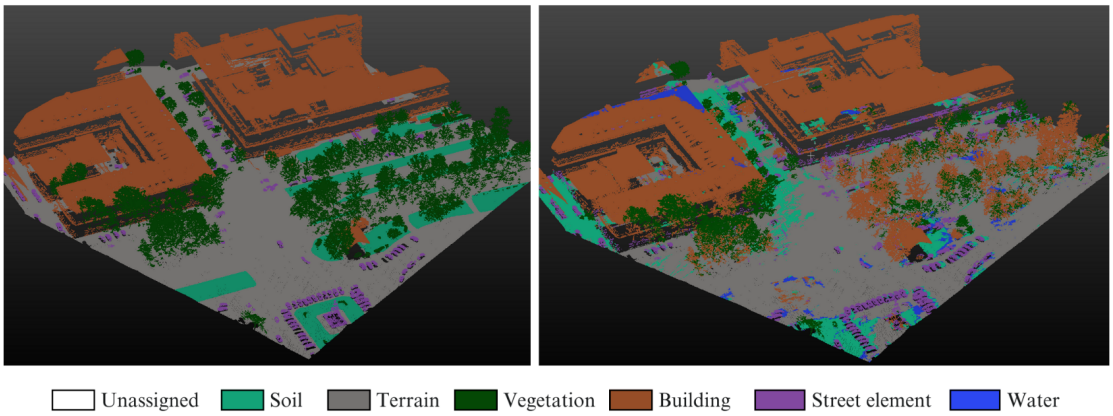



**Figure 5.2:** An example of test point clouds from the Turin3D dataset (left), with the corresponding output from the best-performing transfer learning model (right), which uses RandLA-Net and augmentation on the selected datasets, including Sensat, chosen for subsequent experiments.

On the other hand, the deterioration in performance for the Soil class, as shown

in Figure 5.2, is due to the model's difficulty in distinguishing this class properly, likely caused by variations in color and lighting conditions.
After analyzing the results of this experiment, it can be concluded that the best-performing model is RandLA-Net with augmentation and $\mathcal{D}_{Sel}$. For this reason, it will be the only model used in the following experiments.

## 5.4 Semi-Supervised Learning

As previously illustrated, the transfer learning experiments demonstrated gaps in learning some specific classes. After selecting RandLANet with augmentation as the best architecture, semi-supervised learning approaches were explored to further improve performance on Turin3D, generating pseudo-labels on the unlabelled train set using different confidence thresholding strategies. The results obtained for the different semi-supervised experiments can be seen in Table 5.3.

| **Pseudo-Label Thresholding** | **Iteration** | ***Soil*** | ***Terrain*** | ***Vegetation*** | ***Building*** | ***Street Elements*** | ***Water*** | **mIoU** | **F1** |
|---|---|---|---|---|---|---|---|---|---|
| **Same Threshold** for each class | | | | | | | | | |
| Fixed Confidence equals 0.85 | 1 | 0.0 | 40.65 | 78.93 | 79.49 | 25.99 | 6.74 | 38.63 | 57.70 |
| **Different Threshold** for each class | | | | | | | | | |
| Fixed Confidence per 600 epoch | 1 | 4.16 | 38.25 | 86.98 | 77.54 | 17.17 | 5.42 | 38.25 | 47.22 |
| Fixed Confidence per iteration | 1 | 26.17 | 50.26 | 85.38 | 73.94 | 17.77 | 17.88 | 45.23 | 57.67 |
| | 2 | 32.26 | 34.38 | 86.52 | 66.50 | 27.55 | 0.0 | 41.20 | 63.16 |
| | 3 | 26.29 | 32.64 | 68.60 | 55.74 | 7.32 | 0.0 | 31.7 | 51.49 |
| Adaptive Confidence per iteration | Not weighing the loss | | | | | | | | |
| | 1 | 26.17 | 50.26 | 85.38 | 73.94 | 17.77 | 17.88 | 45.23 | 57.67 |
| | 2 | 30.28 | 52.29 | 87.80 | 77.68 | 19.69 | 23.27 | **48.49** | 61.12 |
| | 3 | 32.89 | 50.88 | 87.62 | 69.92 | 18.01 | 30.51 | 48.30 | **74.45** |
| | Weighing the loss | | | | | | | | |
| | 1 | 24.90 | 52.22 | 85.58 | 78.59 | 17.77 | 16.55 | 45.93 | 57.88 |
| | 2 | 24.47 | 52.97 | 85.71 | 76.33 | 19.47 | 19.85 | 46.97 | 59.49 |
| | 3 | 28.97 | 53.73 | 85.62 | 77.65 | 19.98 | 20.41 | 47.73 | 60.29 |

**Table 5.3:** Results for experiments with Semi-supervised learning with fixed and adaptive confidence per iteration, using RandLA-Net with Augmentations, evaluated on test set of Turin3D ($\mathcal{D}_T^{test}$ ) considering *IoU* per class, *mIoU* and *F1 score*.

The first experiments used a fixed confidence thresholds as illustrated in Section 4.3.1 value equal to 0.85 for all classes. This choice was made based on the assumption that using a relatively high value would provide the most accurate pseudo-labels as possible. Nevertheless, it has been observed that using this threshold emphasizes the imbalance that exists between the classes. In fact the class *Soil(1)* disappears completely. On the other hand, the performance remains high for the classification of *Vegetation (3)* and *Building (4)* (78.93 and 79.49 *mIoU*). As stated in Section 4.3, this is not a good approach since each class has significantly different confidence distributions. For this reason next experiments will use distinct

confidence thresholds for each class.
To improve segmentation accuracy and reduce class imbalance, iterative experiments were conducted using two different approaches for confidence thresholding. In the first approach, the confidence thresholds for each class were kept fixed across all iterations. In the second approach, the thresholds for each class were updated in each iteration, along with the pseudo-labels. Both thresholding methods start with the same result in the first iteration (45.23 *mIoU* and 57.67 *F1 score*), since they use the same as confidence thresholds. This first iteration already represents a substantial 6.50 improvement over the transfer learning technique (38.73 *mIoU*). The difference between the two strategies can be seen from the second iteration. In fact, even the distribution of confidence from one iteration to the next can vary. In the case of the *fixed threshold strategy*, performance progressively deteriorates with each iteration, dropping to 31.76 *mIoU* by the third iteration. Notably, after the first iteration, the *Water (6)* class disappears entirely, reverting to the initial situation where all classes used the same confidence threshold. This supports the idea that each time the model performs inference on the training dataset, the confidence distribution changes, necessitating an update of the thresholds accordingly. In contrast, when the confidence thresholds are updated iteratively, a consistent improvement is observed. The *mIoU* reaches 48.49 in the second iteration before stabilizing at 48.30 in the third iteration, while the F1 score increases significantly to 74.45. Particularly significant is the improvement in *Water (6)* segmentation, which increases from 17.88 to 30.51 *IoU* (+12.63) across iterations. *Vegetation (3)* maintains high *IoU* values around 87, while *Soil (1)* shows steady improvement, reaching 32.89 at the third iteration. Thus, it can be observed that the classes that were the most problematic obtain significant improvements from using this approach, where both the pseudo-labels and the confidence thresholds are recalculated iteratively in each iteration. To improve performance, an additional experiment was conducted in which the loss was weighted by confidence, following the same iterative approach described above. The formula used for this approach is given by Equation (4.4), which can be found in Section 4.3. The results obtained from this method also show a similar trend. In this case, improvements are more gradual, with *IoU* values increasing linearly between iterations( in the first the *mIoU* is 45.93, in the second 46.97, in the third 47.73). Notably, the performance of the model improves steadily at each step, reflecting the more refined adjustments made during training as a result of using loss-weighting the confidence of each pseudo-label. All classes show stable improvements across different iterations with respect to the previous experiments.
In conclusion, both strategies demonstrate very similar performance, with the first strategy yielding slightly higher results. However, the second may be preferred due to its greater stability, as it does not exhibit the fluctuating trends seen in the first.

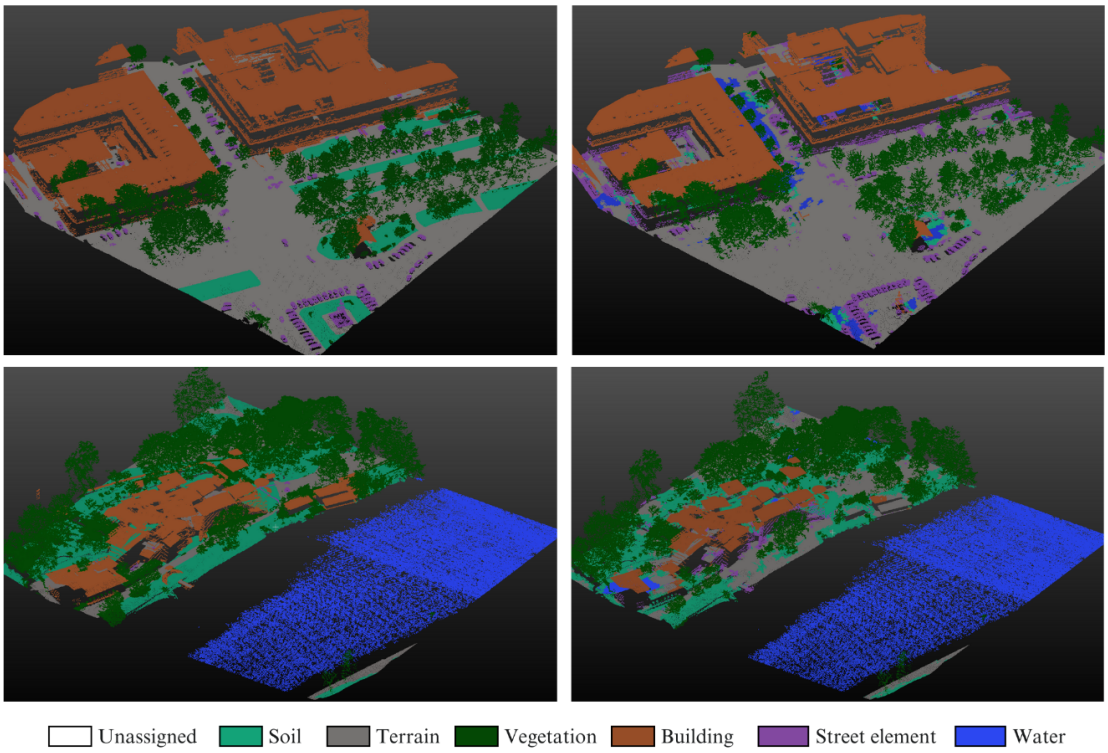



**Figure 5.3:** An example of test point clouds from the Turin3D dataset (left), with the corresponding output from the best-performing model on the Turin3D test subset (right), which utilizes iteratively updated confidence thresholds.

In general, classes *Building (4)* and *Terrain (2)* showed fluctuations in performance with both approaches, demonstrating the difficulty in generating consistent pseudo-labels for these classes that may have a complex and diverse appearance even within the same dataset. On the other hand, results for class *Water (6)* highlight the effectiveness of the adaptive approach, in fact performance raised during the various iterations, unlike the fixed approach that made this class disappear totally.
In conclusion, a significant improvement can be seen using semi-supervised techniques (48.49 $mIoU$) compared to using transfer learning techniques (38.73 $mIoU$). An overall gain of +9.76 demonstrates the significant value of adaptive pseudo-labeling strategies for point cloud segmentation tasks in urban context. Furthermore, for the sake of completeness, an additional experiment was conducted considering a long training loop lasting 600 epochs without confidence thresholds updates instead of iterative updates. This was done to rule out the possibility that the improvement came from simply training for 600 epochs instead of using the iterative approach with 200 epochs for each of the three iterations. However, as shown by the results, training for 600 epochs without the iterative updates led to much

lower performance (*mIoU* of 38.35 and *F1-score* of 47.22) compared to the iterative approach. Therefore, it can be concluded that the improvements are not due to the longer training duration but rather the iterative approach itself. Improvements come from recalculating pseudo-labels, which occurs even when fixed thresholds are used. For this reason, all iterative models perform better than this one where pseudo-labels are no longer recalculated.

# Chapter 6

# Conclusions and Future Works

During this thesis work, a new dataset, Turin3D, was introduced, containing data collected through LiDAR sensors during aerial flights over the San Salvario district in Turin, Italy. The dataset covers an area of approximately $1.43km^2$ and represents a significant contribution to the field of 3D semantic segmentation. It captures a diverse range of urban environments, from green parks to dense urban centers, bridges, rivers, and hilly terrains. The data were acquired using the Leica CityMapper-2 system, which combines LiDAR and high-resolution optical imagery, resulting in a colorized 3D point cloud that maintains both geometric accuracy and radiometric consistency. A key feature of the Turin3D dataset is the well-defined semantic label taxonomy, which includes six main classes: Unassigned, Soil, Terrain, Vegetation, Building, Street elements, and Water. These labels were carefully chosen to be distinct and relevant to urban planning applications, ensuring a high level of heterogeneity between classes and homogeneity within them. The dataset was split into training, testing, and validation sets, manual annotations performed for the validation and test subsets. This dataset, with more than 69 million points, is a valuable resource for future research and applications in urban planning, 3D modeling and semantic segmentation, providing a rich basis for understanding complex and diversified urban environments.
In this work, Transfer Learning and Semi-Supervised Learning techniques were explored for semantic segmentation of urban point clouds using the Turin3D dataset. Several state-of-the-art architectures were evaluated, applying data augmentation strategies to improve generalization. The experiments showed that RandLA-Net with augmentation achieved the best performance, reaching 34.57 $mIoU$ on Turin3D. Including SensatUrban in the source datasets further enhanced generalization, leading to 38.73 $mIoU$. To address class imbalances, semi-supervised learning with

adaptive pseudo-labeling was investigated, resulting in a significant improvement, with 48.49 $mIoU$ (+9.76 compared to transfer learning). The iterative update of confidence thresholds proved to be the most effective approach, particularly benefiting challenging classes such as Water and Soil. These results highlight the advantages of adaptive pseudo-labeling in improving point cloud segmentation and underline the importance of domain adaptation techniques for better generalization across urban environments.

In particular, these techniques enabled effective use of the unlabeled training set, highlighting their potential. Given that, in this field, fully annotated datasets are often unavailable due to the time-consuming and costly nature of annotation, leveraging these approaches can offer performance improvements. The results obtained in this research are promising; however, there is certainly room for improvement in future studies. Firstly, having a fully annotated Turin3D dataset would allow for supervised training, enabling a more accurate evaluation of the techniques used. Additionally, further experiments could be conducted using newer neural network architectures. For instance, in this research, only Point Transformer v1 (PTv1) was used, while the literature also offers two more recent versions (PTv2 and PTv3). Future studies could explore these architectures or other state-of-the-art models to assess whether different architectural choices could provide additional benefits. Another possible approach is to explore domain adaptation techniques specifically designed for 3D point cloud semantic segmentation. These methods could improve model generalization by reducing the impact of the varying environmental characteristics of different cities. This would eliminate the need for fully annotated datasets, effectively addressing the initial challenge.

# Bibliography

[1] P. M. Mather. *Computer Processing of Remotely-Sensed Images: An Introduction*. USA: John Wiley & Sons, Inc., 2004 (cit. on p. 6).

[2] F. P. Medina, L. Ness, M. Weber, and K. Y. Djima. «Heuristic Framework for Multiscale Testing of the Multi-Manifold Hypothesis». In: *Research in Data Science*. Springer, 2019, pp. 47–80 (cit. on p. 6).

[3] F. Patricia Medina and Randy C. Paffenroth. «Machine Learning in LiDAR 3D point clouds». In: *CoRR* (2021). URL: https://arxiv.org/abs/2101.09318 (cit. on p. 6).

[4] Huang Zhang, Changshuo Wang, Shengwei Tian, Baoli Lu, Liping Zhang, Xin Ning, and Xiao Bai. «Deep learning-based 3D point cloud classification: A systematic survey and outlook». In: *Displays* (Sept. 2023), p. 102456. URL: http://dx.doi.org/10.1016/j.displa.2023.102456 (cit. on pp. 6, 7).

[5] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. «Deep Learning for 3D Point Clouds: A Survey». In: *CoRR* abs/1912.12033 (2019). arXiv: 1912.12033. URL: http://arxiv.org/abs/1912.12033 (cit. on pp. 7–9, 14).

[6] 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: *CoRR* abs/1911.11236 (2019). arXiv: 1911.11236. URL: http://arxiv.org/abs/1911.11236 (cit. on pp. 8, 11, 32, 37, 48).

[7] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. «PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation». In: *CoRR* abs/1612.00593 (2016). arXiv: 1612.00593. URL: http://arxiv.org/abs/1612.00593 (cit. on pp. 8–10).

[8] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. *Pointwise Convolutional Neural Networks*. 2018. arXiv: 1712.05245 [cs.CV]. URL: https://arxiv.org/abs/1712.05245 (cit. on p. 9).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385 (cit. on p. 12).

[10] Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. *Dilated Point Convolutions: On the Receptive Field Size of Point Convolutions on 3D Point Clouds*. 2020. arXiv: 1907.12046 [cs.CV]. URL: https://arxiv.org/abs/1907.12046 (cit. on p. 12).

[11] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. *Point Transformer*. 2021. arXiv: 2012.09164 [cs.CV]. URL: https://arxiv.org/abs/2012.09164 (cit. on pp. 12, 48).

[12] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. *KPConv: Flexible and Deformable Convolution for Point Clouds*. 2019. arXiv: 1904.08889 [cs.CV]. URL: https://arxiv.org/abs/1904.08889 (cit. on p. 13).

[13] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. *3D Semantic Segmentation with Submanifold Sparse Convolutional Networks*. 2017. arXiv: 1711.10275 [cs.CV]. URL: https://arxiv.org/abs/1711.10275 (cit. on pp. 14, 37, 49).

[14] Qingyong Hu, Bo Yang, Sheikh Khalid, Wen Xiao, Niki Trigoni, and Andrew Markham. *Towards Semantic Segmentation of Urban-Scale 3D Point Clouds: A Dataset, Benchmarks and Challenges*. 2021. arXiv: 2009.03137 [cs.CV]. URL: https://arxiv.org/abs/2009.03137 (cit. on pp. 15, 29, 32, 36, 37, 53).

[15] Weixiao Gao, Liangliang Nan, Bas Boom, and Hugo Ledoux. «SUM: A benchmark dataset of Semantic Urban Meshes». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 179 (Sept. 2021), pp. 108–120. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2021.07.008. URL: http://dx.doi.org/10.1016/j.isprsjprs.2021.07.008 (cit. on pp. 16, 29, 32, 36, 37).

[16] Iaroslav Melekhov, Anand Umashankar, Hyeong-Jin Kim, Vladislav Serkov, and Dusty Argyle. *ECLAIR: A High-Fidelity Aerial LiDAR Dataset for Semantic Segmentation*. 2024. arXiv: 2404.10699 [cs.CV]. URL: https://arxiv.org/abs/2404.10699 (cit. on pp. 17, 29, 32, 37).

[17] Charles Gaydon, Michel Daab, and Floryne Roche. *FRACTAL: An Ultra-Large-Scale Aerial Lidar Dataset for 3D Semantic Segmentation of Diverse Landscapes*. 2024. arXiv: 2405.04634 [cs.CV]. URL: https://arxiv.org/abs/2405.04634 (cit. on pp. 18, 29, 32, 36, 37).

[18] Weikai Tan, Nannan Qin, Lingfei Ma, Ying Li, Jing Du, Guorong Cai, Ke Yang, and Jonathan Li. «Toronto-3D: A Large-scale Mobile LiDAR Dataset for Semantic Segmentation of Urban Roadways». In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2020, pp. 797–806. DOI: 10.1109/cvprw50498.2020.00109. URL: http://dx.doi.org/10.1109/CVPRW50498.2020.00109 (cit. on pp. 19, 29, 32, 36, 37).

[19] Meida Chen, Qingyong Hu, Zifan Yu, Hugues Thomas, Andrew Feng, Yu Hou, Kyle McCullough, Fengbo Ren, and Lucio Soibelman. *STPLS3D: A Large-Scale Synthetic and Real Aerial Photogrammetry 3D Point Cloud Dataset*. 2022. arXiv: 2203.09065 [cs.CV]. URL: https://arxiv.org/abs/2203.09065 (cit. on pp. 20, 29, 32, 36–38).

[20] Gülcan Can, Dario Mantegazza, Gabriele Abbate, Sébastien Chappuis, and Alessandro Giusti. *Semantic Segmentation on Swiss3DCities: A Benchmark Study on Aerial Photogrammetric 3D Pointcloud Dataset*. 2020. arXiv: 2012.12996 [cs.CV]. URL: https://arxiv.org/abs/2012.12996 (cit. on pp. 21, 29, 32, 37).

[21] Michael Kölle, Dominik Laupheimer, Stefan Schmohl, Norbert Haala, Franz Rottensteiner, Jan Dirk Wegner, and Hugo Ledoux. «The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo». In: *ISPRS Open Journal of Photogrammetry and Remote Sensing* 1 (Oct. 2021), p. 100001. ISSN: 2667-3932. DOI: 10.1016/j.ophoto.2021.100001. URL: http://dx.doi.org/10.1016/j.ophoto.2021.100001 (cit. on pp. 22, 29, 32, 37).

[22] Alperen Enes Bayar, Ufuk Uyan, Elif Toprak, Cao Yuheng, Tang Juncheng, and Ahmet Alp Kindiroglu. *Point Cloud Segmentation Using Transfer Learning with RandLA-Net: A Case Study on Urban Areas*. 2023. arXiv: 2312.11880 [cs.CV]. URL: https://arxiv.org/abs/2312.11880 (cit. on pp. 23, 37).

[23] «Semi-supervised point cloud segmentation using self-training with label confidence prediction». In: *Neurocomputing* 437 (2021), pp. 227–237. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.01.091. URL: https://www.sciencedirect.com/science/article/pii/S0925231221001739 (cit. on p. 24).

[24] Zhimin Chen, Longlong Jing, Liang Yang, Yingwei Li, and Bing Li. *Class-Level Confidence Based 3D Semi-Supervised Learning*. 2022. arXiv: 2210.10138 [cs.CV]. URL: https://arxiv.org/abs/2210.10138 (cit. on pp. 24, 25).

[25] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. *Point Transformer V3: Simpler, Faster, Stronger*. 2024. arXiv: 2312.10035 [cs.CV]. URL: https://arxiv.org/abs/2312.10035 (cit. on p. 37).

[26] Xidong Peng, Xinge Zhu, and Yuexin Ma. *CL3D: Unsupervised Domain Adaptation for Cross-LiDAR 3D Detection*. 2022. arXiv: 2212.00244 [cs.CV]. URL: https://arxiv.org/abs/2212.00244 (cit. on p. 38).

[27] Alhassan Mumuni and Fuseini Mumuni. «Data augmentation: A comprehensive survey of modern approaches». In: *Array* 16 (2022), p. 100258. ISSN: 2590-0056. DOI: https://doi.org/10.1016/j.array.2022.100258. URL: https://www.sciencedirect.com/science/article/pii/S2590005622000911 (cit. on p. 45).

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. on p. 49).