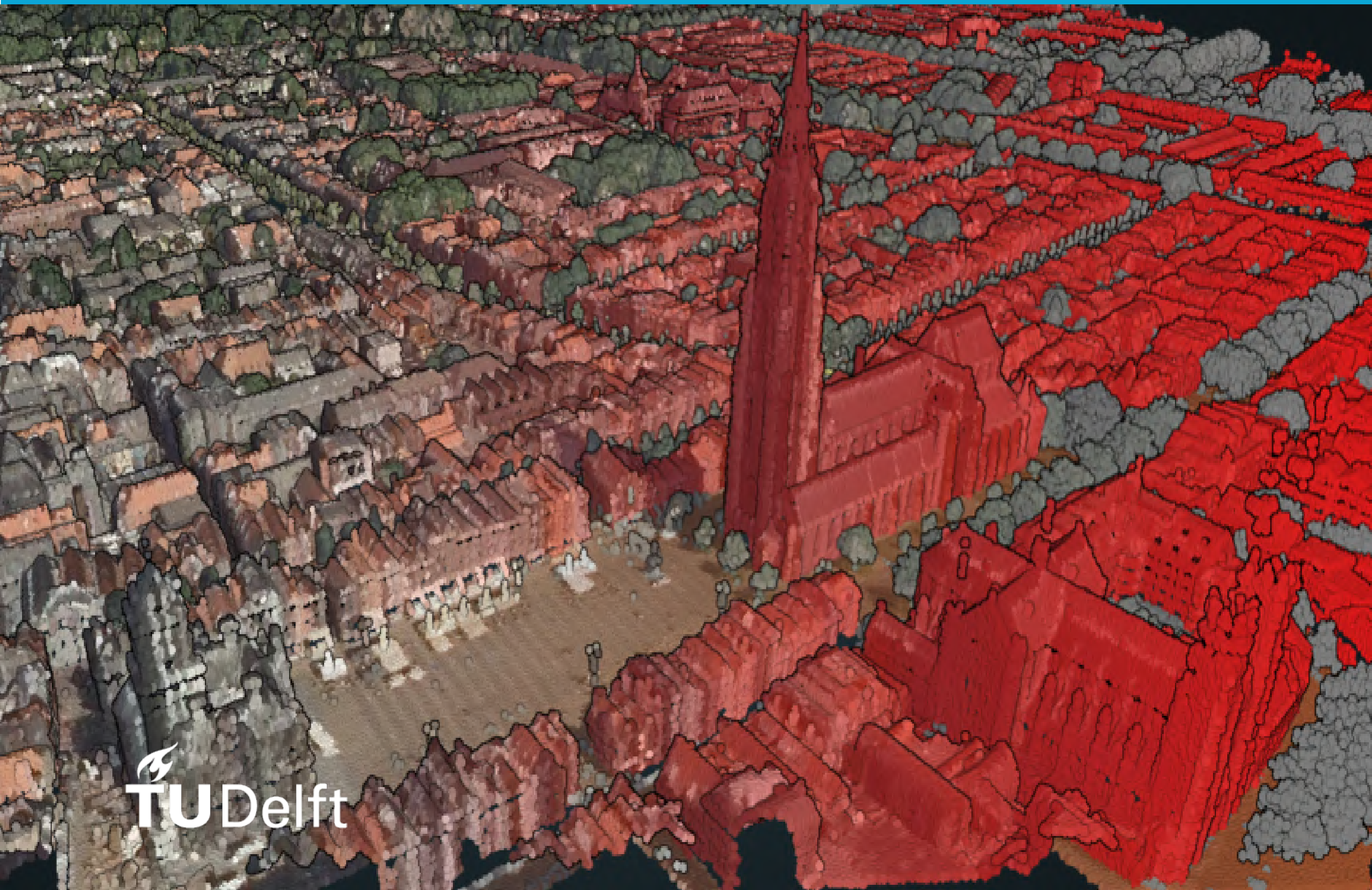
MSc thesis in Geomatics

A Confidence-aware Deep Learning Framework for Refining Laser-scanned Point Cloud Classification

Sharath Chandra Madanu

2024



MSc thesis in Geomatics

A Confidence-aware Deep Learning Framework for Refining Laser-scanned Point Cloud Classification

Sharath Chandra Madanu

2024

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

Sharath Chandra Madanu: *A Confidence-aware Deep Learning Framework for Refining Laser-scanned Point Cloud Classification* (2024)

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Shenglan Du
	Daan van der Heide
	Jantien Stoter
Co-reader:	Shiming Wang

Abstract

Accurately classifying laser-scanned point cloud data remains a critical challenge in geospatial analysis, particularly due to the complexity and volume of the data. This thesis presents a novel, confidence-aware deep learning framework designed to improve the classification accuracy of point cloud data, specifically focusing on the Actueel Hoogtebestand Nederland (AHN) dataset. The framework integrates geospatial knowledge into the deep learning process, enabling the model not only to refine its predictions through iterative learning but also to enhance the training data along the way via iterative online learning, ensuring continuous improvement in both training data quality and model performance.

The preprocessing phase assigns confidence scores to each point in the point cloud based on local neighborhood properties, with additional input from multispectral imagery (MSI) to further enhance the confidence estimation. These confidence scores are central to the online learning process, where the model prioritizes high-confidence points for training while progressively updating lower-confidence points to improve accuracy. To test the hypothesis that confidence-aware learning can enhance point cloud classification, we selected the KPConv network due to its suitability for handling unstructured data and capturing complex geometric features.

Extensive experiments demonstrate that the proposed framework, particularly with the *Online* strategy, enables deep learning models to perform better when trained solely on native point cloud attributes (elevation and intensity) compared to models without this strategy. Importantly, the *Online* strategy qualitatively enhances the training data by refining labels and reducing noise, thereby supporting more robust model performance. While incorporating additional features from aerial imagery showed no overall improvement, specific classes, like *High tension* and *others* did see performance gains.

Github repo: https://github.com/AutumnMoon00/RefineNet

Acknowledgements

This thesis has been a result of several months of research, including both learning and doing. Several people assisted me in the course of this rewarding journey and to all of them I express my sincere thanks.

Each time I got stuck with a paucity in the project, or lost sight of the end when being overwhelmed by small details, Shenglan, my first supervisor, held my back and showed the way. Her support extended beyond technical supervision and helped me propel through phases of self-doubt. You have devoted a lot of time discussing the ideas and guiding me. The way the thesis started, shaped up, and concluded, at every step, you were there helping me improve it. Such a smooth journey and learning curve wouldn't have been possible without you. A big thank you!

I would like to thank Daan for your invaluable insights and ideas, which contributed greatly to the results and conclusions of this thesis. I also appreciate your thorough review of my writing, providing clear and actionable feedback. A special thanks to Dr. Jantien Stoter for your steadfast support during all the pivotal moments.

My special thanks to Geodelta, the company I interned at and later worked part-time. Working here being my first industry experience, always remains a dear experience. In a way, my thesis journey has roots here. Months spent overseeing point cloud data quality control, combined with my growing interest in deep learning, helped shape this thesis. I am also grateful for the flexible working culture and financial support, which eased some of my personal stress during this period.

Sometimes, using words to describe deeply felt emotions only dilutes them——or perhaps I'm just not great at expressing them. Whatever the reason, my love and gratitude for my parents, my brother, and my family knows no bounds. This would not have been possible without them. These same feelings extend to my dear friends; nights passed in conversations and music. Being with them is as joyful as seeing my thesis results for the first time.

Sharath

Delft, October 2024

Contents

List of Figures

List of Tables

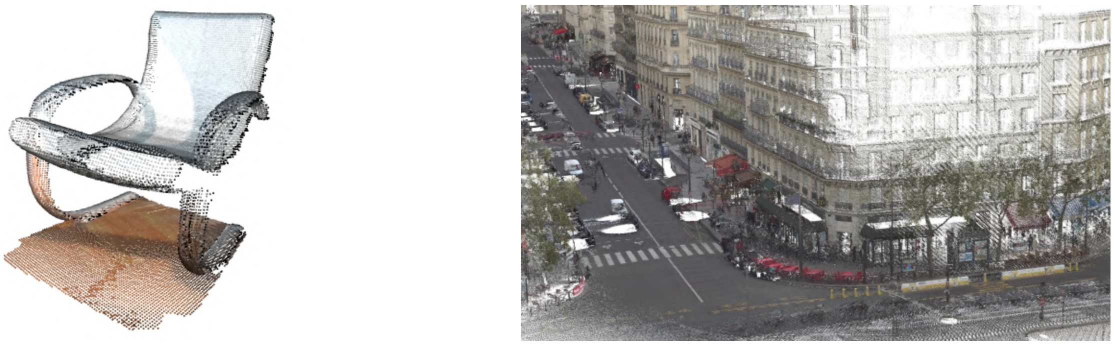
Acronyms

DEM	digital elevation model	12
DSM	digital surface model	1
DTM	digital terrain model	1
LiDAR	Light Detection and Ranging	7
ToF	Time-of-Flight	7
MSI	Multispectral Imagery	5
NIR	Near-Infrared	xi
SWIR	Short-wave Infrared	11
NDVI	Normalized Difference Vegetation Index	11
BAG	Basisregistraties Adressen en Gebouwen	1
AHN	Actueel Hoogtebestand Nederland	1
DNN	deep neural network	19
CNN	convolution neural network	ix
ML	machine learning	3
CRF	Conditional Random Fields	13
DL	deep learning	25
MLP	multilayer perceptron	ix
FPS	farthest point sampling	14
NLP	natural language processing	16
KNN	k-nearest neighbors	17
ERF	effective receptive field	17
MGA	Multi-Granularity Attention	17
GAN	generative adversarial network	4
ALS	Airborne Laser Scanning	xi
TLS	Terrestrial Laser Scanning	8
MLS	Mobile Laser Scanning	8

1 Introduction

Over the past two decades, point cloud data has emerged as a valuable source of rich information, finding wide-ranging applications in diverse fields including floodplain management, robotics, autonomous navigation, and medical imaging. Extracting knowledge from it is driving both the industrial and academic research [Poux and Billen, 2019].

Point cloud can be understood as dense surveying data. Specifically, it can be defined as a collection of data points in a three-dimensional coordinate system {X, Y, Z}. Thus, point cloud is a discretized representation of the world around us. Additionally properties such as color (RGB), intensity, return number, classification could also be stored. These points with properties can be used to create a digital representation of an object or a scene (Figure 1.1).



(a) Point cloud of a chair (source: Open3D) (b) Point cloud of an outdoor scene (source: NPM3D)

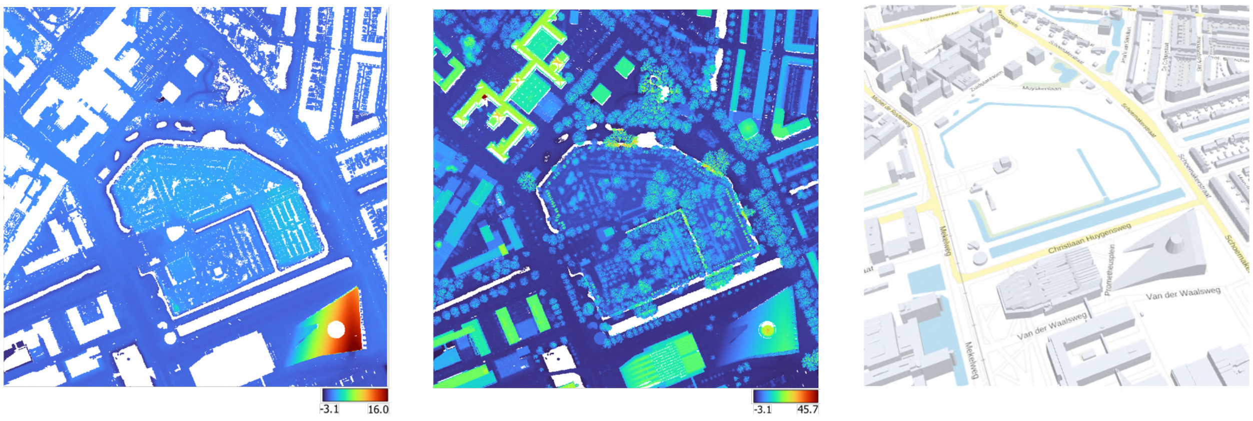
Figure 1.1: Point cloud at different scales

Unlike images, that store information in a 2D gridded structure, point clouds inherently contain 3D spatial data, providing depth information that images lack. This advantage makes them valuable for applications where understanding the shape and structure of objects or environments is crucial, like in autonomous vehicles, architecture, robotics etc [Xie et al., 2020; Zhang et al., 2023]. Further, point clouds can be viewed from any angle, making it possible to analyze an object/scene comprehensively.

In many applications, understanding details in the scene and context is important. This detailed understanding is provided by semantic segmentation of point clouds, where every individual point in the cloud is assigned a class label. These semantically labeled point clouds are essential for developing downstream applications, especially in tasks involving interaction with the environment and decision-making. For instance, in robotics, it is necessary to differentiate between objects like pedestrians, vehicles, and roads for safe and effective operation.

Point cloud data serves as a foundation for creating a range of downstream applications. For instance, in the Netherlands, the elevation model of the country—Actueel Hoogtebestand Nederland (AHN) point cloud data—supports the creation of digital terrain model (DTM), and digital surface model (DSM) raster maps. Specifically, points classified as *'ground'* are used to produce the DTM, while the DSM is derived from all points except those classified as *'water'*. Additionally, *3D BAG*[1], a comprehensive and up-to-date dataset of detailed 3D building models of the Netherlands, is automatically generated by combining building data from Basisregistraties Adressen en Gebouwen (BAG) with height data from AHN.

[1]3D BAG - https://3d.bk.tudelft.nl/projects/3dbag/

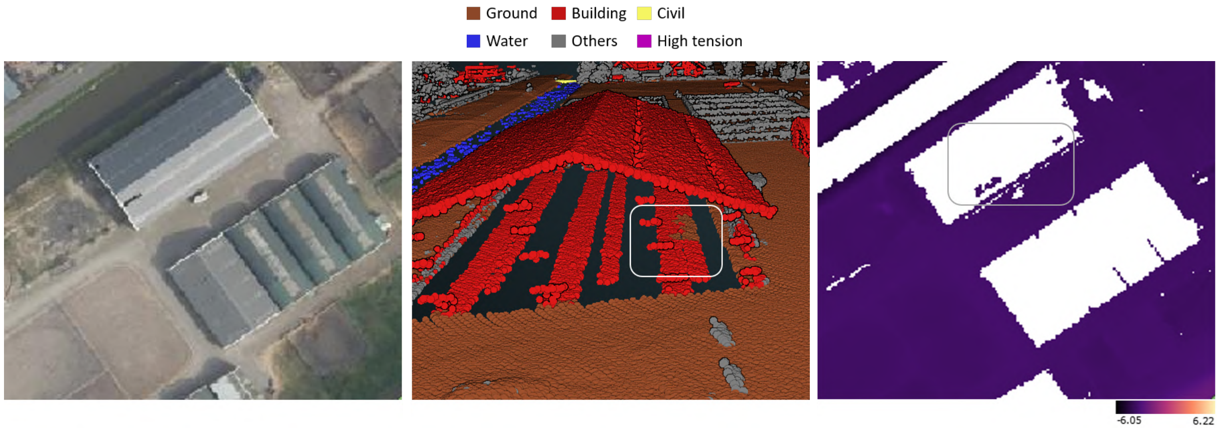


(a) DTM (b) DSM (c) 3D BAG

Figure 1.2: Downstream applications from AHN point cloud

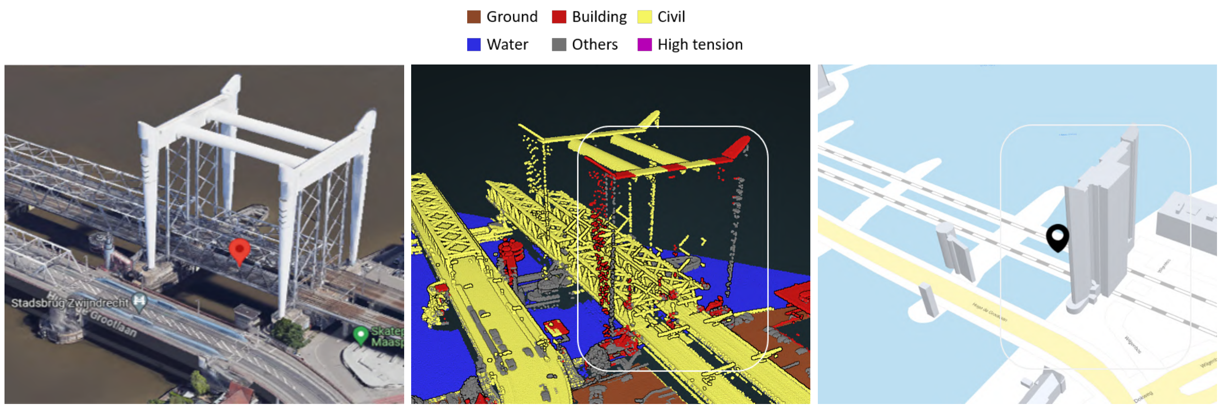
Although 3D point cloud semantic segmentation is crucial for accurate analysis, errors are still quite common. In the AHN data, these segmentation errors can significantly affect the quality of the resulting products. Misclassified points can lead to incorrect interpretations and flawed models. For example, *'ground'* points incorrectly classified as being inside *'buildings'* can affect the DTM, leading to inaccurate representations where the ground surface is shown within the building footprints (Figure 1.3). Such errors can lead to unrealistic expectations for further analysis, such as water flow modeling, as they may incorrectly suggest the presence of pathways through buildings. Similarly, the misclassification of a *'bridge'* as a *'building'* results in an inaccurate representation of structures in the 3D BAG dataset (Figure 1.4). These type of errors can propagate through various applications, ultimately reducing the reliability of the derived products.

Noisy classifications further exacerbate these issues, as illustrated in Figure 1.5, where *'ground'* points are mixed with *'other'* points, resulting in inconsistencies. Figure 1.6 shows how misclassifying a jetty, which should be labeled as *'civil structure'*, with *'ground'* noise impacts the DTM. For additional examples of label errors and inconsistencies in AHN data, please refer to my internship report with Geodelta [Madanu, 2024].



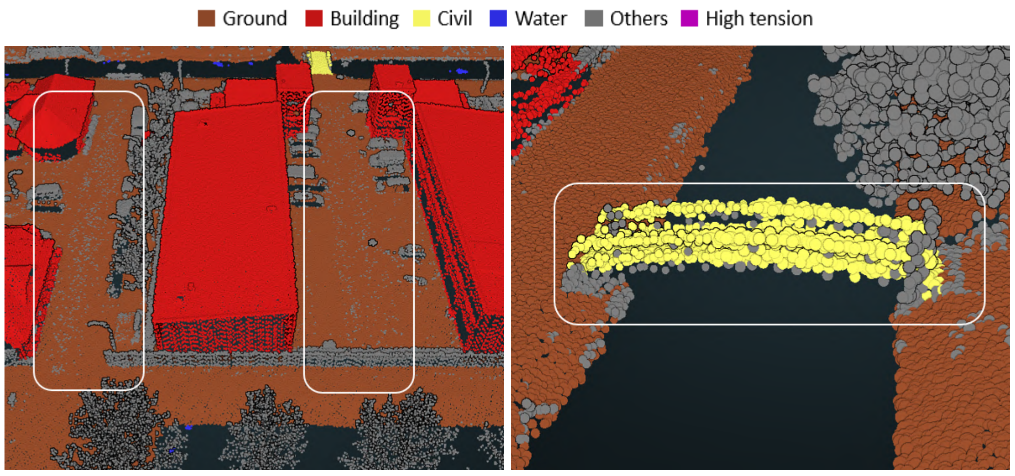
(a) RGB (b) AHN4 point cloud side view (c) DTM

Figure 1.3: Ground points inside building and reflected in DTM



(a) RGB (b) AHN4 point cloud (c) DTM

Figure 1.4: Bridge misclassified as building and its reflection in 3D BAG



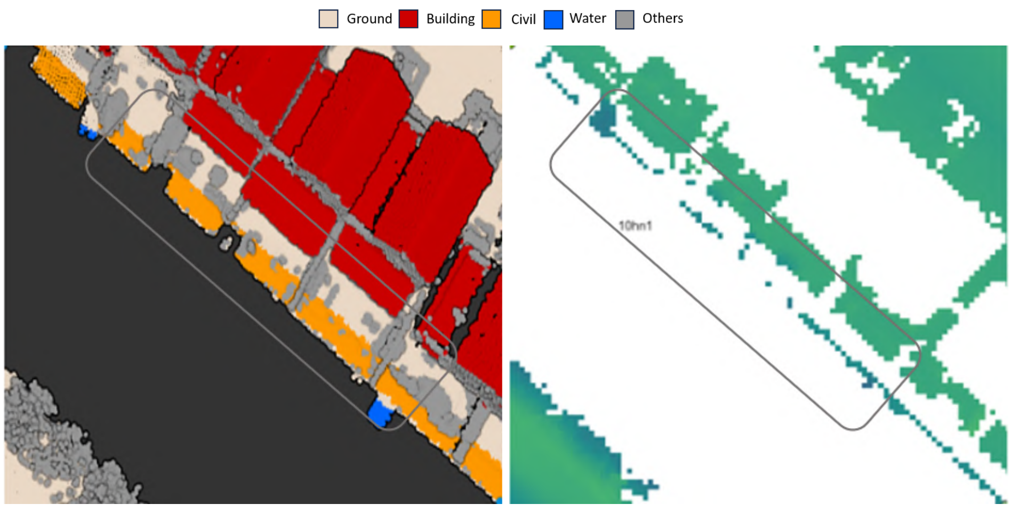
(a) *'Ground'* mixed with *'other'* (b) *'Civil'* mixed with *'other'*

Figure 1.5: Noisy classifications

Semantic segmentation of the point clouds typically begins with either manual annotation or semi-automated approaches, where pre-trained models provide initial labels that are then refined manually. To relieve the burden of manual annotation, various machine learning (ML) methods have been developed, generally following a pipeline of *feature extraction, feature selection, and classification* [Weinmann et al., 2015]. Prominent methods ML classification include Random Forests [Grilli et al., 2019], Support Vector Machines (SVM) [Zhang et al., 2013], and Conditional Random Fields (CRF) [Niemeyer et al., 2012]. These traditional ML techniques often face challenges due to their dependence on handcrafted features, which require domain expertise, and also their limited ability to capture complex spatial relationships inherent in dense point cloud data.

Deep learning, on the other hand, has demonstrated significant advantages over traditional ML approaches by eliminating the need for manual feature engineering. Since the introduction of *PointNet* in 2016 [Qi et al., 2016], a sequence of deep learning methods specifically designed for 3D point cloud semantic segmentation has been developed. These methods take raw point clouds as input and can be broadly categorized into multi-layer perceptron (MLP)-based architectures, convolutional neural networks (CNNs), and transformer-based models. These advancements allows deep learning models to autonomously learn high-dimensional features, eliminating the need for manual feature engineering, making them more suitable for complex tasks in geomatics and 3D computer vision [He et al., 2021].

Despite these advancements, deep learning models remain vulnerable to data quality issues; their performance is highly dependent on the quality of the input data. Good quality input yields good quality predictions. While several data-efficient approaches have been proposed to mitigate data quality and



(a) RGB (b) DTM

Figure 1.6: Jetty (*'Civil'*) classified with *'Ground'* noise and its reflection on DTM

scarcity issues, they often involve complex models like generative adversarial network (GAN) and require extensive training cycles [Li et al., 2021]. Therefore, there is a need for a simpler yet effective method to improve model performance, particularly in correcting misclassifications in point cloud data.

This thesis proposes a deep learning framework coupled with an *Online* training label update strategy for point cloud semantic segmentation that incorporates geospatial knowledge as priors. First, we give a point-wise confidence measure to estimate to what extent we can trust the given class label. Then, by iteratively refining misclassified points during training, the proposed approach enhances both the quality of the training data and the model's overall performance. This method aims to address the limitations of existing techniques by providing a straightforward solution that improves segmentation accuracy without the complexity of more intricate models.

The key contributions of this thesis are as follows:

1. **Confidence measurement strategy**: Developed a method to evaluate the quality of existing semantic labels by combining local neighborhood consistency with geometrical knowledge derived from additional data sources.
2. **Customized deep Learning framework**: A deep learning framework adapted for segmenting the Dutch AHN point cloud, utilizing a dynamic selection of the most confident training samples.
3. **Label update mechanism**: Designed a label refinement strategy that iteratively improves training data quality, thereby boosting the performance and robustness of teh deep learning model.

1.1 Research objective

The main research goal of this thesis project is to:

Develop a deep learning framework that automatically improves the existing classifications of laser-scanned point cloud data by correcting misclassifications.

To achieve this, the following sub-questions will be relevant:

1. How can geospatial knowledge be incorporated into a deep learning framework, and what benefits does this integration provide?
2. To what extent does the online learning strategy enhance the model's ability to correct misclassifications and improve overall segmentation accuracy compared to traditional training approaches?

3. What is the impact of incorporating additional spectral features (such as NIR and RGB) on the performance of the proposed confidence-aware deep learning model for point cloud segmentation?

1.2 Scope

With the current technology in the industry, point clouds can be generated at various scales; starting from internal body organs at mm level, to country level dataset. The approaches in this thesis are developed for outdoor with AHN4 data as reference, and all the data integrated into the projected are also massive and available at country level. So its applicability, if any, on varied datasets at different scales, such as indoor point cloud data or body parts is not known. Further, With change in meaning of what counts as *'building'* and *'high-tension'* from AHN4 to AHN5, the model trained on one dataset may not be usable on the other.

1.3 Thesis Outline

This thesis is organized into six chapters. Chapter 2 provides the background on point clouds, aerial Multispectral Imagery (MSI), and relevant deep learning methods for point cloud segmentation. Chapter 3 outlines the research methodology, detailing the confidence measurement, online deep learning framework, and evaluation metrics. Chapter 4 explains the geographical context of the research and implementation aspects, including data sources, software, and hardware used. Chapter 5 presents the experimental results, comparing models trained with various features and evaluates the impact of online learning during the training process. Finally, Chapter 6 discusses the findings, addressing the research questions and suggesting directions for future work.

2 Background

This chapter introduces the fundamental concepts and technologies relevant to this research. It covers point cloud fundamentals, including acquisition methods and key properties, followed by an exploration of the AHN dataset and multispectral imagery (MSI). Finally, the challenges associated with point cloud classification are examined, along with a review of state-of-the-art machine learning and deep learning methods used for point cloud segmentation, setting the stage for the methods developed in this thesis.

2.1 Point cloud fundamentals

Point cloud could be understood as dense surveying data. Specifically, point cloud can be defined as a collection of data points in a three-dimensional coordinate system (x, y, z). Thus point cloud is discretized representation of the world around us. Additional properties such as color (RGB), intensity, return number, classification could also be stored. The points represent the external surface of an object or scene, capturing its shape, possibly its color, and other attributes. Point clouds have varied applications in a range of fields such as computer graphics, virtual reality, 3D modeling, and autonomous navigation.

As discussed in Introduction, point clouds have tremendous advantages over 2D images, however there are also a few disadvantages:

1. **Large data size and complexity:** Point clouds can consist of millions or even billions of points, resulting in large data sizes requiring significant storage and computational resources to process and analyze [Cai, 2024].
2. **No surface connectivity:** Point clouds inherently do not describe information between points. Additional processing is required to create continuous surfaces.
3. **No color and texture information:** Unlike 2D images, piont clouds typically do not capture color or texture information unless additional sensors or techniques are used.

2.1.1 Acquisition methods

Point clouds could be generated from a variety of methods like 3D scanners, and photogrammetry. Photogrammetry creates point cloud of a scene by combining overlapping images captured from multiple directions. This is done by identifying and matching common points (features) across multiple images, followed by optimizing 3D positions of the camera and the features. Matched features are used to create a dense point cloud, representing the 3D structure of the scene.

Light Detection and Ranging (LiDAR), a 3D scanning technology, is an active remote sensing technique that emits intense, focused beams of light and measures the time it takes for the reflections to be detected by the sensor. There are two kinds of range measurements in scanning systems: Time-of-Flight Time-of-Flight (ToF) and phase shift based systems. Today, most scanning systems employ ToF ranging principle [Fernandez-Diaz et al., 2014]. Using this system, the time for the laser to make round trip to and from the reflective surface is measured. Combined with knowledge of the speed of light in the medium between the sensor and the reflecting surface, the distance d between them is calculated (Eq. 2.1 and Figure 2.1, where c = *speed of light, and* t = *time-of-flight between light emitted and detected*).

$$d = \frac{c * t}{2} \tag{2.1}$$

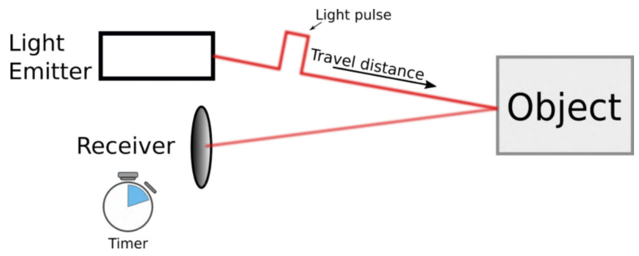


Figure 2.1: Time-of-Flight ranging measurement (source: http://tof-insights.com/)

For terrestrial mapping, a laser in the near-infrared spectrum range (850-940 nm) is used [Mazzari, 2019]. This is one of the main disadvantages, where raw point cloud in itself does not contain RGB information. A camera sensor along with LiDAR has to collect information, and later in processing, data from both the sensors has to be fused [Yao et al., 2024].

LiDAR systems primarily employ two methods for capturing return signals: discrete return and full waveform. Discrete return LiDAR records specific points where the laser pulse is reflected back to the sensor, typically capturing a finite number of returns. This method simplifies data processing and reduces data volume but may miss subtle details. In contrast, full waveform LiDAR records the entire return signal as a continuous waveform. This method provides a more detailed representation of the scanned area by capturing the complete energy profile of the returned pulse. Today, most of the LiDAR system stores data in discrete form, and the return number is stored along with the xyz information.

LiDAR data can be collected using several acquiring techniques, namely Airborne Laser Scanning (ALS), Terrestrial Laser Scanning (TLS), Mobile Laser Scanning (MLS), and Handheld (Backpack) Laser Scanning. The main difference is that TLS, MLS and handheld LiDAR techniques are measured from ground, whereas airborne is from air. TLS is a static method in which LiDAR is mounted on top of a tripod and for MLS the scanner is mounted on a vehicle. Static methods are commonly preferred for construction sites and archaeology, while mobile technique for road mapping, infrastructure assessment, and city modeling. Handheld LiDAR allows for flexible data collection and data collection in hard-to-reach areas like forests. Typically airborne technique is used for mapping large areas like landscapes, forests, and urban environments.

ALS systems are equipped with GPS/IMU units on board, that helps in tracking platform's positions and altitudes, then ranges measured at these points allows their ground elevation values to be determined [Jie Shan, 2018] (Figure 2.2).

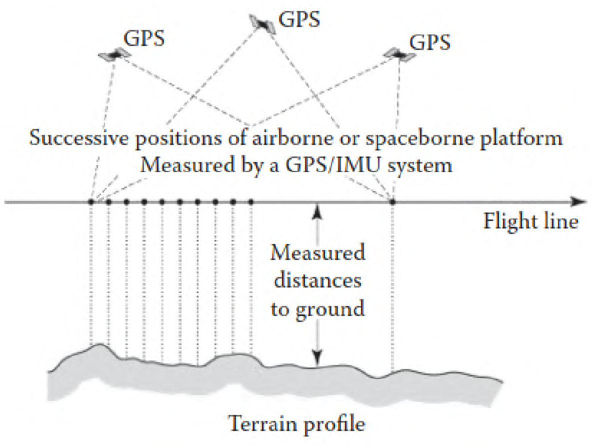

GPS
GPS
GPS
Successive positions of airborne or spaceborne platform
Measured by a GPS/IMU system
Flight line
Measured distances to ground
Terrain profile

Figure 2.2: ALS measuring terrain profile [Jie Shan, 2018]

ALS is carried out in two dimensions. The first dimension follows the direction of the aircraft, achieved through its forward motion. The second dimension, typically perpendicular to the flight path, is accomplished by a scanning mechanism. Common designs for this mechanisms are oscillating mirror, rotation polygon, and Palmer scanner. The flight course combined with the scanning mechanism creates various patterns of how points are detected on the ground [Fernandez-Diaz et al., 2014] (Figure 2.3).

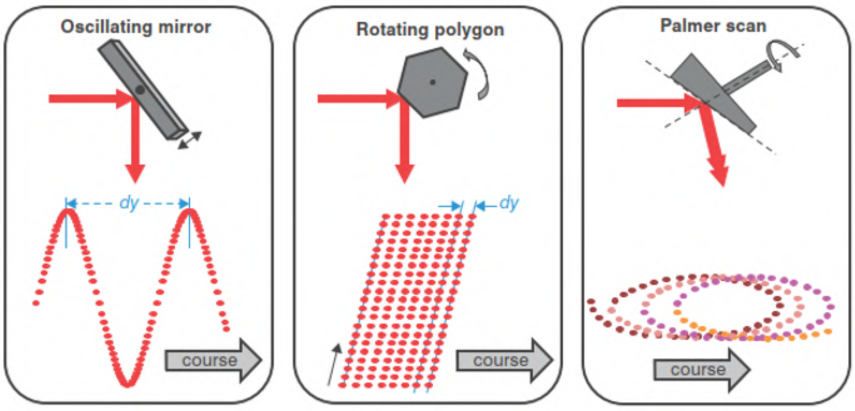
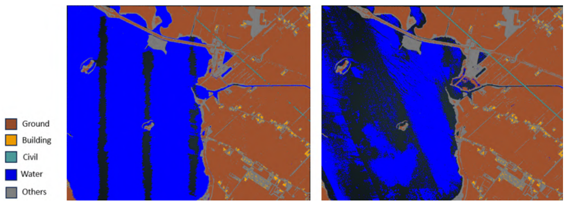


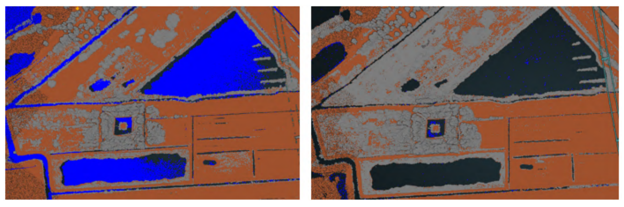
Figure 2.3: Ground track patterns of different scanning mechanisms

The Palmer scan produces an elliptical scanning pattern on the ground. This allows each point on the ground to be scanned twice from different angles (forward and backward views). Scanners that use a rotating polygon produce a unidirectional, constant-velocity scan in a regular raster pattern over the ground [Petrie, 2011].

Water surfaces absorb and reflect LiDAR pulses differently than terrestrial surfaces. This often results in lower point returns from water surfaces. The rotating polygon scanning method detects more water surface due to its varied angles of incidence. When the flight path is directly above the water, this scanner sends laser pulses at multiple angles, increasing the likelihood of capturing returns from the water surface. Whereas with Palmer scanning, is less effective over water. Even when the flight path is directly overhead, the scanner's fixed angle results in many pulses reflecting away from the sensor, reducing data capture from the water surface. AHN4 project used a scanner which implements rotating polygon scanning, and for AHN5 project data is being collected using Leica CityMapper - 2 that implements Palmer scanning technology. Its effect on water data collection in AHN is shown in Figure 2.4



(a) whole tile



(b) Zoomed-in view

Figure 2.4: Comparing AHN4 and AHN5 (tile 16AN1)

2.1.2 Components of point cloud

In the context of point cloud analysis, properties can be broadly divided into primary and secondary components [van der Heide et al., 2024; Donkers, 2024]. The primary components focus on the core aspects of the point cloud, such as coverage, relative accuracy, and absolute accuracy. These are crucial for tasks like accurate spatial measurements and feature extraction, as they determine the overall quality and usability of the data for foundational analysis and decision-making.

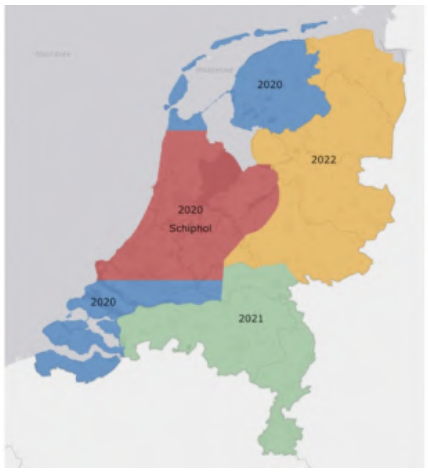
However, this thesis is concerned primarily with the secondary components, which include features like RGB coloring, intensity values, and classifications. While secondary components may not directly influence the spatial accuracy of the point cloud, they play a significant role in applications that rely on enhanced data interpretation, such as semantic classification and visualization. In particular, this study emphasizes the role of RGB and NIR information in improving point cloud segmentation accuracy, even though it is not part of the primary geometric properties. By leveraging these secondary aspects, the thesis aims to explore how they can enhance the model's ability to differentiate between different classes, thus improving the overall effectiveness of point cloud classification.

2.2 Actueel Hoogtebestand Nederland (AHN)

Actueel Hoogtebestand Nederland (AHN) is the height model of the Netherlands. It is a digital dataset that provides information about the elevation and topography of the Dutch landscape. AHN makes various data sets available, such as point clouds in LAZ format, and raster images in GeoTIFF format for Digital Terrain Model (DTM) and Digital Surface Model (DSM) at resolutions of 0.5 m and 5.0 m. It is widely used in industry for various projects. As a considerable portion of the land lies below sea level, the AHN plays a crucial role in flood risk management and water resource planning.

The measurements for the first AHN program were made in 1996. From then on the AHN program has evolved from AHN1 to AHN4, providing the complete elevation datasets of the whole country [AHN, 2024a]. Currently the latest, AHN5, is in data collection phase, and only a part of the Netherlands's data is made available.

The AHN4 data was measured between 2020 and 2022 (Figure 2.5a). For the AHN4, the point density will be about 10-14 points per square meter. For the area around Schiphol this can be 20-24 points per square metre. The point cloud data is provided in LAZ format which is a compressed LAS file file format. LAZ point clouds are heavy files and therefore most users use derivative products [AHN, 2024b].



(a) Data collection timeline [AHN, 2024a]

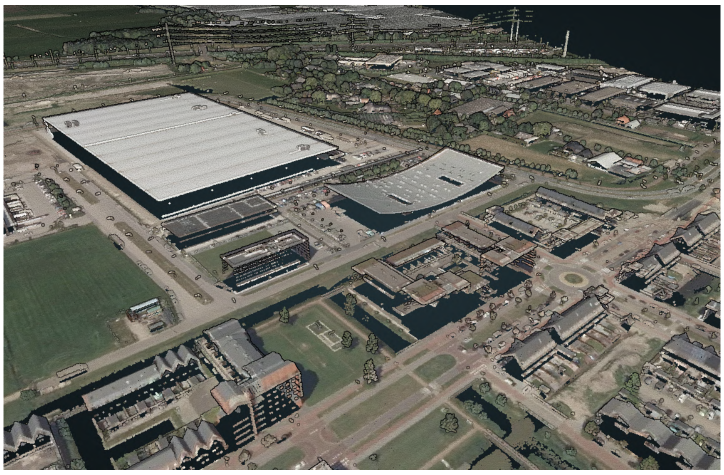

(b) AHN4 colored point cloud

Figure 2.5: AHN4 acquisition timeline and colored point cloud

The AHN point cloud data has six classes of labels, and their definitions are mentioned in Table 2.1. Concerning the semantic segmentation of the point cloud there are a few changes in the meaning given to points over the versions of AHN:

1. Prior to AHN5, all the points of the building, including walls, roofs, windows, sunshades, etc, are given building classification. However, with AHN5, only the points that reflected from the roof are classified as buildings, and the remaining are of classification others. ***add a picture***

2. In AHN4, only the high-tension cables are given the classification of high-tension, but the towers that hold them are of classification others. Coming to AHN5, both the cables and towers are labelled as high-tension. ***add a picture***

Label	Definition
0	never classified
1	other
2	ground
6	building
9	water
14	high-tension
26	civil structure (bridges and jetties)

Table 2.1: AHN point cloud label values and definitions

2.3 Multispectral imagery (MSI)

Multispectral imagery refers to capturing images across multiple bands of the electromagnetic spectrum, where each band represents a specific range of wavelengths. Unlike standard color photography that captures light in only three bands (red, green, and blue). Multispectral images capture data from a wider range of the spectrum, including visible light, NIR, and Short-wave Infrared (SWIR). MSI provides more detailed information about objects and surfaces than standard color imagery [Schowengerdt, 2007].

Each material reflects radiation differently across these bands, creating a unique pattern. These unique patterns in which surfaces reflect energy are called ”spectral signatures”, and can be used to identify materials based on their interaction. The data collected forms a graph or a signature, plotting reflectance or emission against wavelength (Figure 2.6).

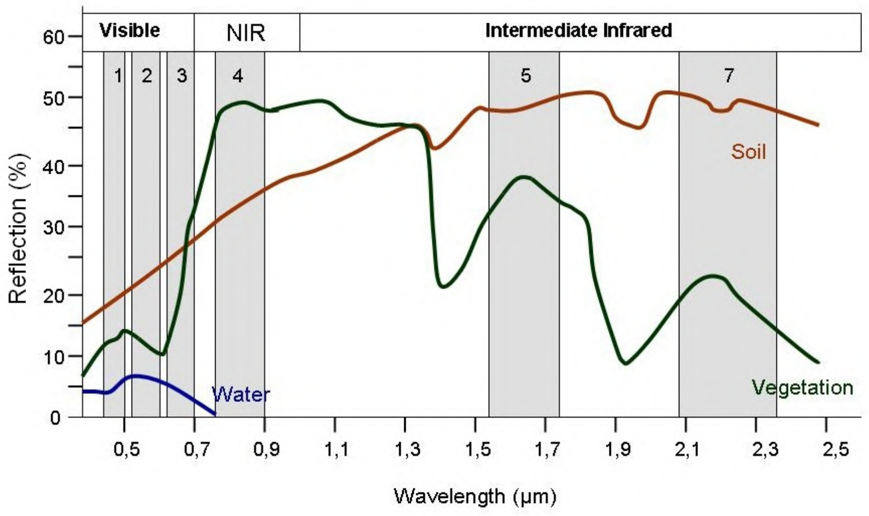


Figure 2.6: Spectral signatures of soil, vegetation and water [Siegmund and Menz, 2005]

2.3.1 Normalized Difference Vegetation Index (NDVI)

The Normalized Difference Vegetation Index (NDVI) is a numerical indicator used to analyze remote sensing measurements and assess whether the target being observed contains live green vegetation. NDVI is calculated using the visible red and NIR bands of the electromagnetic spectrum (Eq 2.2). Vegetation typically reflects more NIR and absorbs more visible light, making Normalized Difference Vegetation Index (NDVI) a useful measure for vegetation health and and density [Pettorelli, 2013; Xie et al., 2008].

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{2.2}$$

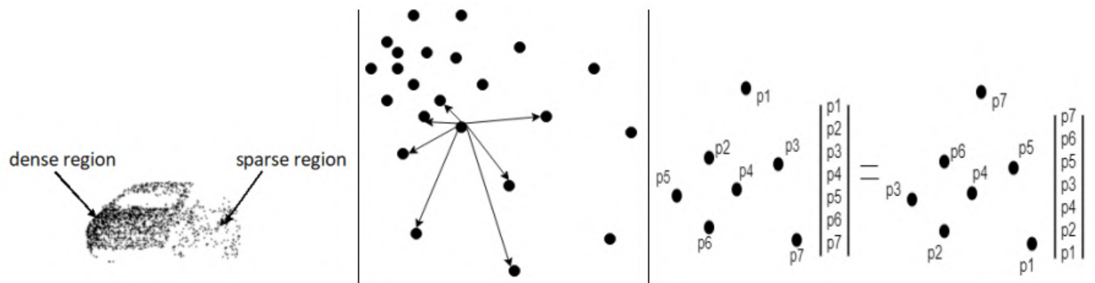
While NDVI is primarily used to map vegetation, it will only provide limited information about the built environment. This is because buildings and other non-vegetative surfaces typically have low NDVI values, similar to water bodies and bare soil, as they lack the photosynthetic activity of vegetation. The general steps to delineate buildings from NDVI data:

1. Obtain high-resolution NDVI data for the study area using satellite imagery.
2. Combine it with additional data, such as LiDAR, digital elevation model (DEM), or other spectral indices, to improve the accuracy of building delineation [Elshehaby and Taha, 2009].
3. Apply a threshold to the NDVI values to identify areas with log vegetation (e.g.NDVI < 0.3).
4. Use image segmentation techniques, such as edge detection or object-based image image analysis, to group the low-NDVI pixels into distince building footprints.
5. Post-process the building footprints to remove small artifacts and refine the boundaries.

2.4 Challenges with point cloud

It is essential to examine some key properties of point cloud data that are relevant for automatic semantic segmentation of point clouds. Certain challenges stem from the inherent qualities of point cloud, as proposed by Bello et al. [2020], and they are

- **Irregularity (data sparsity and density variation)**: Point clouds can be highly irregular, with varying densities across different areas. Dense areas may capture and represent intricate object features, while sparse areas might lack sufficient information (Figure 2.7).
- **Unstructured**: Point cloud data data lack a regular grid arrangement. Each point is scanned individually, and the distances to neighbouring points can vary. Which is unlike images, where adjacent pixel-pixel distance is fixed.
- **Unorderedness**: A point cloud represents a scene using a collection of points (usually represented by XYZ coordinates). These points are typically stored as a list in a file [Bello et al., 2020]. Importantly, the order in which the points are stored does not alter the scene representation; meaning, it remains invariant under permutations [Qi et al., 2016, 2017].



(**a**) Irregular. Sparse and dense regions. (**b**) Unstructured. No grid; each point is independent and the distance between neighboring points is not fixed. (**c**) Unordered. As a set, point clouds are invariant to permutation.

Figure 2.7: Point cloud data challenges (Bello et al. [2020])

2.5 Machine learning for point cloud segmentation

Traditional methods for point cloud segmentation typically rely on geometric features and spatial relationships to group points into meaningful segments. These approaches include clustering algorithms like k-means; region growing methods where segments expand from seed points based on similarity criteria; and model-fitting techniques such as RANSAC. These methods focus on dividing the point cloud into non-overlapping regions by grouping points based on simple, hand-crafted features like curvature, color, normal vectors, and smoothness, as well as geometrical constraints [Zhan and Yu, 2012; Rabbani Shah et al., 2006]. However, since these methods do not incorporate supervised prior knowledge, the resulting segments lack semantic information.

Supervised machine learning techniques addresses this problem. Machine learning way can be described in a stepwise approach, Figure 2.8 [Weinmann et al., 2015]:

1. **Neighborhood selection:** The first step is to select the local neighborhood around each 3D point. Typically the neighborhood is selected either by a fixed radius (spherical/cylindrical) [Thomas et al., 2018] or k-nearest neighbors.
2. **Feature extraction:** For each point from its neighborhood various geometric features are extracted. These features could include measures like curvature, normal vectors, and other shape descriptors that capture local geometry around a point.
3. **Feature selection:** Number of features can quickly go out of hand, so it is crucial to select only the most meaningful ones. This step reduces the dimensionality of the data, and improves efficiency and accuracy of the classification process.
4. **Classification algorithm:** Finally, classification algorithms such as Random Forests [Grilli et al., 2019], Support Vector Machines (SVM) [Zhang et al., 2013], Conditional Random Fields (CRF) [Niemeyer et al., 2012], etc. are applied on extracted features. These algorithms learn to categorize points into specific classes based on features extracted and the contextual information provided.

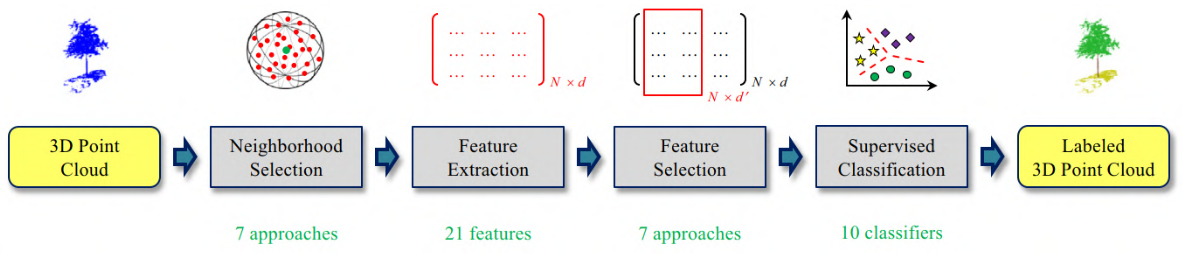


Figure 2.8: Machine learning point cloud semantic segmentation framework proposed by Weinmann et al. [2015]

Following the above steps, each point is given label based on its own individual features. Further, neighborhood selection also helps measuring just the geometrical features, but fails to encode contextual information. This leads to unavoidable noisy classification. To this extent, Niemeyer et al. [2012] proposed using Conditional Random Fields (CRF) which adds contextual information into the model. Landrieu et al. [2017] proposed a framework that takes the labeled points from above network and combines them with a graph-based contextual information. This process is referred to as *structured regularization* or *smoothing*, which tries removing noise and smoothens the labels.

2.6 Deep learning for point cloud segmentation

Unlike conventional and ML approaches, deep learning methods automatically extract the features of the point cloud and achieve better performance [Zhang et al., 2023]. Deep learning based methods for semantic segmentation of point clouds can be categorized into two categorized into two categories: *projection-based* and *point-based* methods. These techniques address the intrinsic challenges presented by point clouds—primarily their unorderedness and lack of structure. Projection-based methods approach the challenge of unstructured data by transforming the original point cloud data into structured data like images, voxels, or even graphs. Networks extract features using this transformed data as input. Whereas point-based methods embrace the chaos. They directly work with raw point clouds, taking away the necessity of preprocessing or tranformation into a structured grid.

2.6.1 Multilayer perceptron (MLP)-based methods

MLP is a type of neural network consisting of multiple layers of neurons, with each layer fully connected to the next, typically used for supervised learning tasks like classification and regression.

PointNet [Qi et al., 2016] was the first technique to apply MLP directly to point clouds, and lay foundational principles that have influenced subsequent developments in the field. It handles raw point clouds by treating each point individually, using shared MLPs to extract features. PointNet also uses a technique called max pooling to create a single global feature vector that captures the overall characteristics of the entire point cloud. This global feature is crucial for tasks like semantic segmentation, as it ensures the results are the same no matter the order of the input points. This allows the model to capture the global features of the point cloud without the need for pre-processing or converting the data into a structured grid.

PointNet fails to capture local structure because it does does not consider local dependency between points. To address this and the problem of varying density in the point clouds, [Qi et al., 2017] made improvements to PointNet. PointNet++ consists of sampling layer, grouping layer, and PointNet backbone network. Firstly, *sampling layer* selects a set of points from input points, which defines the centroids of local regions. These set of points are selected using iterative farthest point sampling (FPS) such that they are more distant to the rest of the points. Followed by *grouping layer*, that constructs local region sets by finding "neighbouring" points around the centers. Each neighbourhood group of points is created by selecting all the points within a certain radius from the subsampled point. Each group of points created by the grouping layer is then independently processed by a mini PointNet network, referred to as the PointNet backbone network, to encode local region patterns into feature vectors. Hierarchically this process is repeated, and thereby reducing the point resolution deep into the network. At the last layer of the hierarchy, all features from the different groups are passed through the PointNet layer to extract a global feature vector. Therefore, PointNet++ captures global information by aggregating local contexts hierarchically.

2.6.2 Convolutional neural network (CNN)-based methods

Unlike images, which has data stored in regular grids, point clouds irregularity and unstructuredness, made traditional CNNs inapplicable. So, in an attempt to take advantage of the established CNN architectures, initial research on CNNs focused on converting point clouds into regular grids (voxels in 3D or projection images in 2D) [Maturana and Scherer, 2015; Su et al., 2015; Kanezaki et al., 2016; Riegler et al., 2016]. However, this intermediary data representation in these methods poses several challenges. Challenges such as quantization artifacts and increased computational complexity, especially with high resolution intermediary structures. With image projection techniques, difficulties such as views occlusions, and difficulties in capturing all objects in large-scale scenes using multiview imaging [Zhang et al., 2023] are common. Voxelization techniques proved to be robust in object detection, but semantic segmentation is still not good enough.

As an alternative *point-based* convolution approach suggests applying convolutions directly on points. Unlike projection-based segmentation methods, point-based convolution methods do not rely on any transformation of the input point clouds into a regular grid format. Further, unlike images, where neighboring pixels are easily identifiable, point clouds lack a fixed spatial relationship between points. To overcome this, point convolutions typically involve the following steps:

1. **Neighborhood Selection:** A local neighborhood is defined for each point in the cloud, often using k-nearest neighbors (KNN) or ball queries based on Euclidean distance. The choice of neighborhood size and shape impacts the receptive field and computational complexity.

2. **Feature Extraction:** Features are extracted from each point and its neighbors. Common choices include raw coordinates, RGB color channels, local normals, or higher-level geometric descriptors. These features are often encoded into a hihger-dimensional space using MLPs.

3. **Weighting and Aggregation:** Through convolution operations, learned weights are applied to the encoded features of each point and its neighbors. The weighted features are then aggregated using aggregators like sum, max pooling, or more complex attention mechanisms.

A basic hierarchical U-net shaped architecture for segmentation tasks is shown in Figure 2.9.

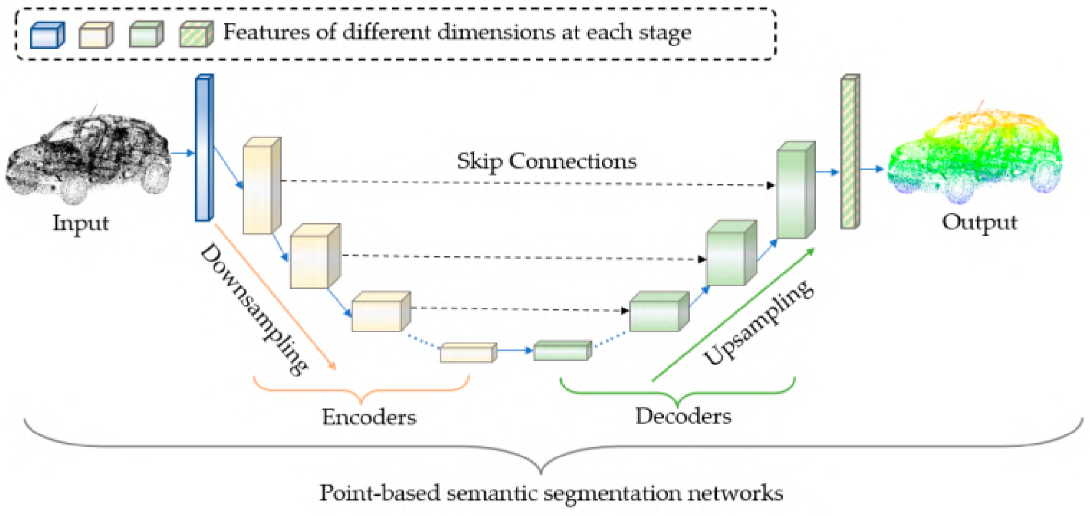

Features of different dimensions at each stage
Input
Skip Connections
Output
Downsampling
Upsampling
Encoders
Decoders
Point-based semantic segmentation networks

Figure 2.9: Basic frameworks of point-based CNNs [Zhang et al., 2023]

Research in this field mainly revolves around the internal structures of encoders and decoders. Specifically how kernel weights are distributed in space, and how convolution on points is defined. Pointwise CNN [Hua et al., 2017] defined kernel weights with voxel bins, Flex-convolution and SpiderCNN [Xu et al., 2018] used linear and polynomial functions to give weights to neighboring points. These approaches had issues like, lack of flexibility due to grid voxels, and functions depended on distance-wise order to assign weights making them spatially inconsistent.

Thomas et al. [2019] proposed a state of the art point convolutional architecture, **KPConv**, which is inspired by image-based convolution, but instead of kernel pixels, kernel points in 3D space are used to define the area where each kernel weight is applied, as shown in Figure 2.10. In images, kernel aligns with the pixels which makes it easy for multiplying kernel weights with pixel feature vector. However, KPConv lacks this flexibility, because kernel points in space do not align with neighbour points. However, KPConv lacks this flexibility, because kernel points in space do not align with neighbour points. Each neighbor point feature vector within the kernel is multiplied by all the kernel weights carried by kernel points, and with correlation factor based on neighbor point's distance from kernel point. For detailed mathematical implementation, please refer to the source KPConv paper [Thomas et al., 2019].

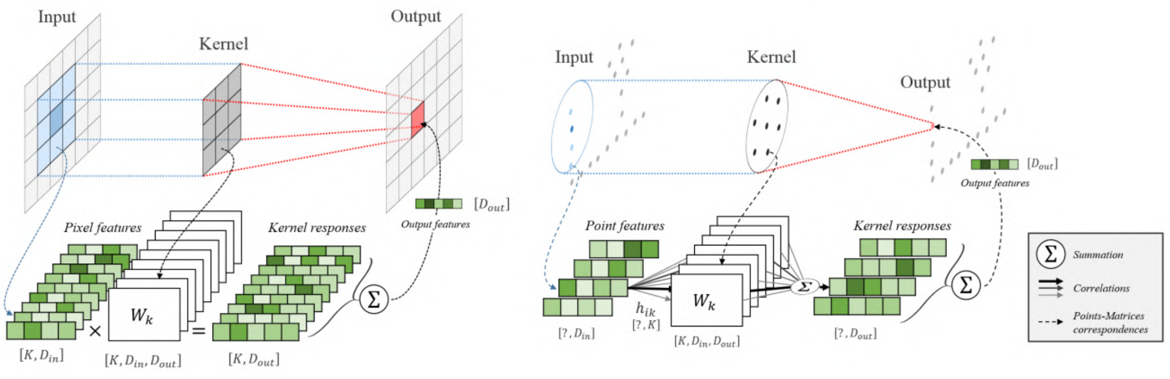


Figure 2.10: Comparison between an image convolution (left) and a KPConv (right) on 2D points. Source: Thomas et al. [2019]

KPConv also follows standard U-shaped architecture (like in Figure 2.9). The detailed source architecture is shown in Figure 3.7.

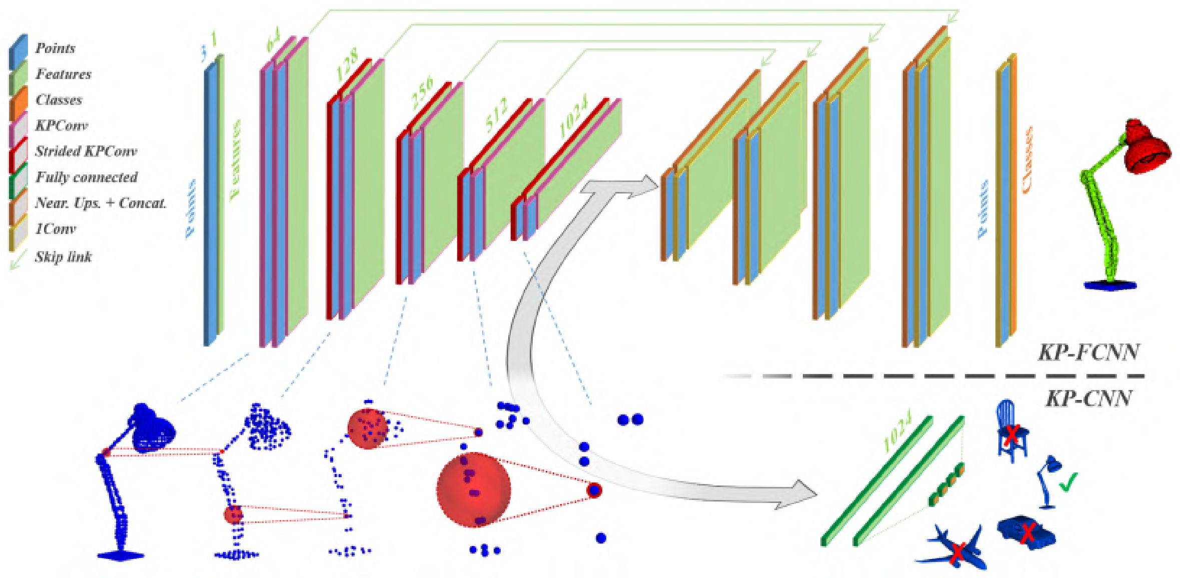


Figure 2.11: KPConv network architecture for segmentation (top) and classification (bottom). Segmentation network is called *KP-FCNN* (fully convolutional network), and an adaptation of it is used for this thesis project. Source: Thomas et al. [2019]

2.6.3 Transformer-based methods

Originally developed for natural language processing (NLP), the architecture was adapted for visual tasks with the introduction of the Vision Transformer (ViT) [Dosovitskiy et al., 2020], and proved to be very successful in 2D image understanding [Parmar et al., 2018; Strudel et al., 2021; Zheng et al., 2020; Sun et al., 2021]. This model demonstrated that transformers could process images not just as arrays pixels, but as sequences of image patches, which allows model to capture dependencies between different parts of an image.

The application of transformers on point cloud is obvious because the self-attention operator, which is at the core of the transformer networks, is a set operator. It is invariant to point set order and cardinality of the input point cloud, which makes it more suitable for point cloud processing than CNN. The first influential paper that uses transformers in this field is *Point Transformer* [Zhao et al., 2020a]. The original idea of transformer model where it is designed for sequence data (like text), utilizing self-attention mechanism to process data points in relation to each other is adopted. The Point Transformer adapts this concept to 3D point clouds by implementing a local neighborhood-based self-attention mechanism where each point in the cloud attends to its neighbours determined by k-nearest neighbors (KNN) approach. The core component of the model is the *point transformer layer*. For which positional encoding plays a crucial role in NLP, and standard way to encode them into features is by using sine and cosine or normalized range values (Vaswani et al. [2017], Zhao et al. [2020a], Zhu et al. [2020]). An advantage with point cloud is that positional information is inherently part of the data (as points occupy specific position in space), and in the point transformer layer relative position between the neighbouring points is used for position encoding.

Stratified Transformer (Lai et al. [2022]) improved the existing transformers by better modeling long-range contextual dependencies. They achieve this by enlarging the effective receptive field using a technique called stratification. The method stratifies the input point cloud into different levels based on spatial hierarchies. At higher stratification level, the points are subsampled using FPS (Qi et al. [2017]). To encode context into point's features, the self-attention mechanism aggregates information densely from nearby points (lower stratification level points) and sparsely from distant points (higher stratification level points). For efficient self-attention implementation, to manage computational complexity, the paper introduced efficient self-attention mechanism called *sparse attention*, where only a subset of points are considered for each attention operation. Contrary to most research where the position of points is considered unnecessary for 3D transformer-based networks because the *xyz* coordinates have already been used as the features(Misra et al. [2021]), the stratified transformer adopted a context-based adaptive relative position encoding scheme (Wu et al. [2021]), and achieved better performance compared to no encoding.

Xu et al. [2022]; Zhang et al. [2021] to enlarge effective receptive field (ERF), the point cloud data is voxelized and attention maps are build using the voxels. However, feature information loss and contextual information loss is incurred because of back projecting voxel features to point features using a devoxelizing operator. To address both issues: loss of information due to devoxelization in self-attention, and limited context size, Zhou et al. [2023] proposed *SAT: Size-Aware Transformer*. SAT is designed to tailor effective receptive fields for objects of different sizes using a novel Multi-Granularity Attention (MGA) scheme and a Re-Attention module. MGA utilizes point-voxel cross attention to build attention maps between points and voxels directly, and it doesn't require a devoxelizing operator. Re-attention module refines the output from the MGA by recalibrating the attention scores based on object size, ensuring that different size objects receive appropriate attention weighting.

2.7 Data efficient methods

Data efficient semantic segmentation of point cloud models are crucial in applications where labeled data is scarce or labeling is costly. These models try making most of the limited labeled data available through various techniques and architectures. Many interesting approaches exists, here are a few important ones:

- **Transfer learning and pre-training:** Model is trained to learn general features from large datasets and finetuned to more specific features from smaller datasets.
- **Self-supervised learning:** These methods do not require labeled data. Self-supervised learning models generate their own labels from within the data by defining a pretext task (Jing and Tian [2019]), like predicting a part of point cloud from another part. Without needing many labeled examples, useful representations could be learnt this way.
- **Semi-supervised learning:** This method combines small amount of labeled data with a large amount of unlabeled data achieving better performance than supervised learning with limited labeled data for training (Bergmann [2023]). Semi-supervised models utilize unlabeled data to better understand the distribution of data and complements the learning process with labeled data.
- **Active learning:** From unlabeled dataset, active learning models selectively prioritises the data which needs to be labeled to have highest the impact to train a supervised model. When unlabeled data is too huge, priority has to be given to labeling particular parts of data in a smart way.

Hu et al. [2021] introduced the Semantic Query Network (SQN), a novel weak supervision technique for 3D point cloud segmentation that dramatically reduces the need for labeled data. This network requires only 0.1% of the labels typically needed by fully supervised models. SQN's architecture includes a *point local feature extractor*, which encodes the raw point cloud into hierarchical latent representations, capturing spatial and geometric context through established networks like PointNet, PointNet++, or RandlaNet. Followed by, *point feature query network*, which propagates sparse labeling information by querying and summarizing features from neighboring points, leveraging the assumption that nearby points share similar semantic information. This innovative approach addresses the data inefficiency challenge, setting a new benchmark in weakly supervised learning for 3D point clouds.

Li et al. [2021] proposed a semi-supervised method that utilizes unlabeled data through a *self-training* process that involves a mechanism for predicting the confidence of labels assigned to the unlabeled data. A GAN architecture is adopted with two components: *Segmentation network* and *Discriminator network* (Figure 2.12).

One network performs the actual task by giving labels' to the points, and another network judges the given label by giving confidence scores. The former network is segmentation network and the latter is discriminator network. Segmentation network performs the actual task of segmenting the point cloud, predicting the labels for each point. Discriminator network judges the confidence of the labels predicted by the segmentation network. Based on discriminator's confidence measure, pseudo labels of the unlabeled point clouds that are more trustable are selected to participate in the network training.

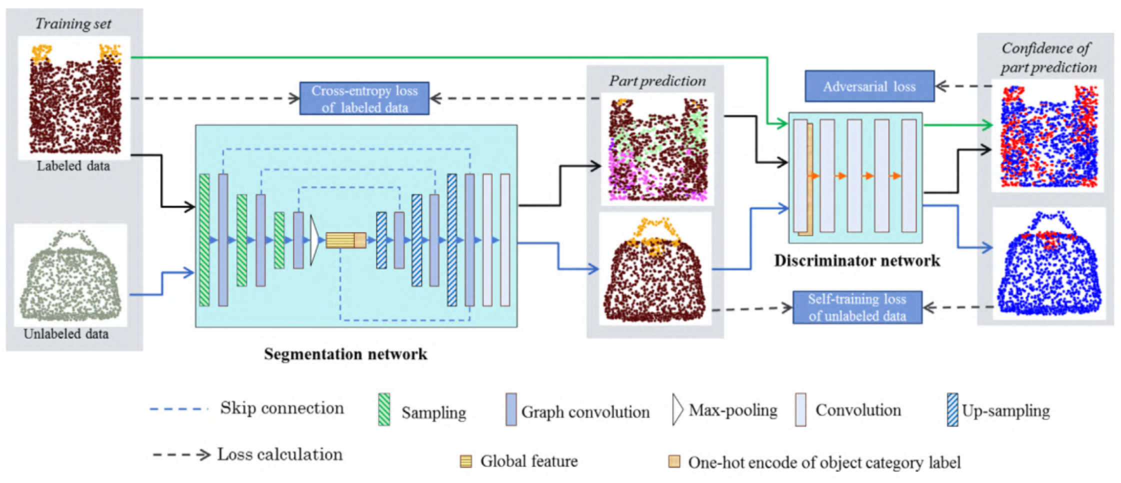


Figure 2.12: Basic frameworks of point-based CNNs [Li et al., 2021]

3 Research method

In recent years, deep neural network (DNN)s have achieved impressive results in 3D point cloud segmentation using supervised learning. However, the effectiveness of these supervised learning techniques heavily relies on the data quality. Poor-quality inputs lead to poor-quality segmentation results, and vice versa. The strong learning abilities of DNNs can become a disadvantage when they learn from incorrect labels, which can impair the model's performance [Ye et al., 2022].

To address this issue, we propose a new deep learning framework that not only learns from accurate labels but also corrects possible mislabelings in the process. Our approach involves two main steps, illustrated in Figure 3.1:

1. **Preprocessing**: We calculate and assign confidence scores to each point in the point cloud based on heuristic knowledge.
2. **Network training**: Confidence values guide the deep learning model to learn from most supposedly correct classifications.

3.1 Preprocessing for confidence measurement

Confidence is how confident we are with the current label of the point. The confidence score ranges from zero to one, with lower values indicating less trust in the current classification and higher values representing greater certainty. During preprocessing, we assign a confidence score to each point to quantify the reliability of its semantic label.

Importantly, confidence is measured with the belief that it helps in segregate points that are likely misclassified or have noisy labels. However, confidence values are not absolute, and should not be viewed as infallible indicators of label accuracy.

Calculating confidence is a two step process. First, *primary confidences* are calculated based on neighborhood consistency. This is followed by a *refinement* process that further adjusts these primary confidences based on heuristic knowledge of the data, such as building footprints.

3.1.1 Primary confidence

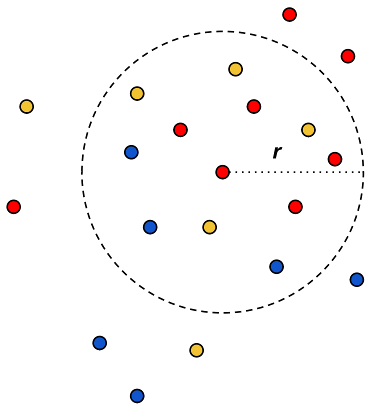
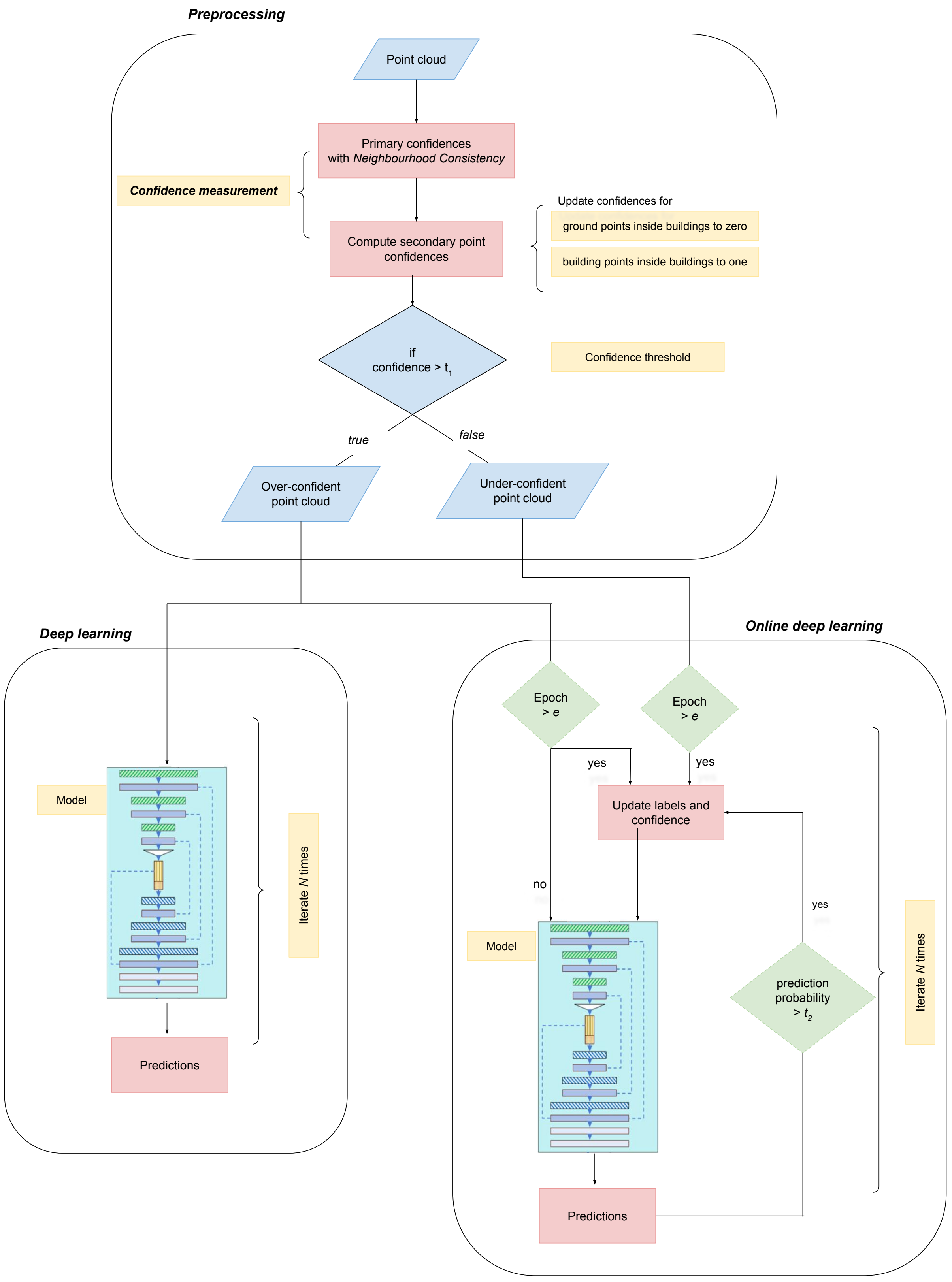


Figure 3.2: Neighbourhood consistency



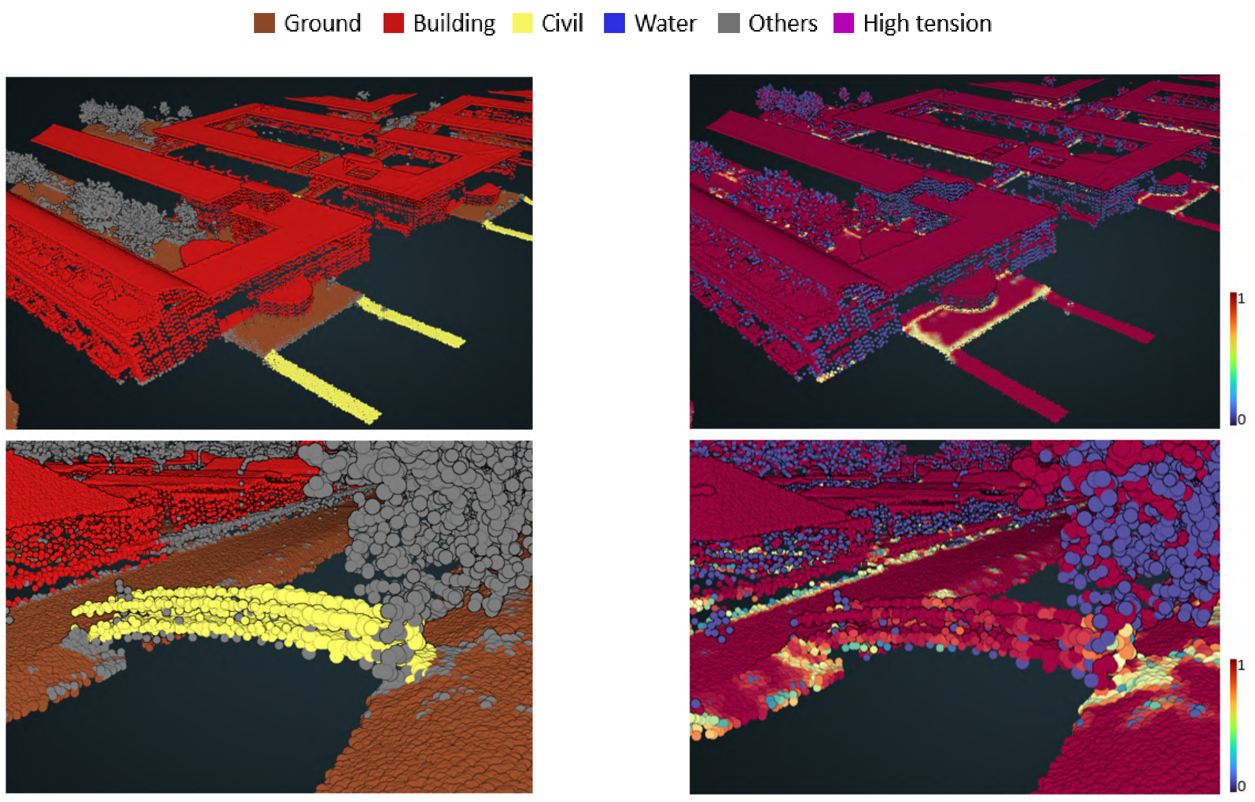
Preprocessing
Point cloud
Primary confidences with Neighbourhood Consistency
Confidence measurement
Compute secondary point confidences
Update confidences for
ground points inside buildings to zero
building points inside buildings to one
if confidence > t_1
Confidence threshold
true
false
Over-confident point cloud
Under-confident point cloud
Deep learning
Online deep learning
Model
Iterate N times
Predictions
Epoch > e
Epoch > e
yes
yes
no
Update labels and confidence
yes
prediction probability > t_2
Model
Iterate N times
Predictions

Figure 3.1: Complete method flowchart

It is fairly logical to assume that points of same classification can be expected together. So, *Neighbourhood Consistency* is measurement of how well a point is surrounded by points of same classification. For each point, within a sphere of radius r surrounding it, the percentage of points sharing the same classification gives the *neighbourhood consistency* confidence score (Figure 3.2). This neighborhood consistency is our primary confidence.

$$C = \begin{cases} \frac{N_{sameclass}}{N_{total}} & \text{if } N_{total} >= 5, \\ 0 & \text{if } N_{total} < 5 \end{cases}$$

where $N_{sameclass}$ is number of points belonging to the same class, and N_{total} is the total number of points including all the classes inside the sphere of radius r. A minimum threshold of five points is chosen within the radius neighborhood. This makes sure there is enough context in the neighborhood, and also eliminates most of the data outliers by giving them zero confidence. In Section 6.1.2, we experiment with thresholds of 5, 10 and 15, showing that a threshold of 5 optimally balances neighborhood density and minimizes the risk of giving high confidence to sparse, and potentially noisy classifications.



(a) Annotated point cloud

(b) Primary confidence

Figure 3.3: Primary confidence by neighborhood consistency measurement

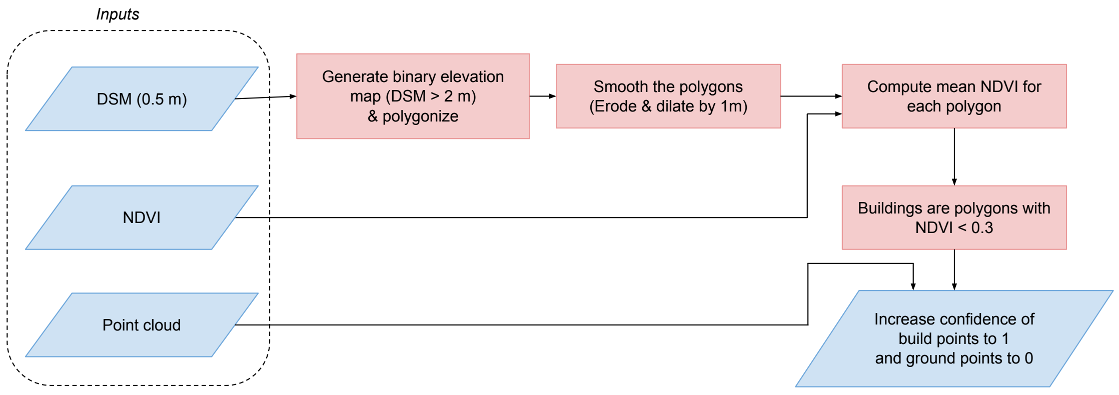

Inputs
DSM (0.5 m)
NDVI
Point cloud
Generate binary elevation map (DSM > 2 m) & polygonize
Smooth the polygons (Erode & dilate by 1m)
Compute mean NDVI for each polygon
Buildings are polygons with NDVI < 0.3
Increase confidence of build points to 1 and ground points to 0

Figure 3.4: Refining confidences through buildings footprint extraction

3.1.2 Refining confidences

In AHN4, it is common to see building outer walls/facades with very less point density, or sometimes no points at all. Together the flight elevation and rotating polygon scanning, as discussed in Chapter 2.1.1, leads to this. In such cases, neighbor consistency fails to give confidence to points because of minimum five points threshold in the radius neighborhood (Figure 3.3).

Since walls and facades are key components that contribute to the overall structure of a building, it's important to enhance their confidence. The first step in achieving this is to identify the building footprint. Afterward, the confidence of the relevant *building* points within this footprint can be increased.

Having building footprints offers an additional advantage: it helps identify an incorrect mislabeling, where *ground* points are mistakenly located inside buildings. These points confidence is reduced to zero. This is critical because if such points are included in the DTM, it could lead to inaccurate surface water modeling, potentially showing water flow through buildings, which is unrealistic.

In this thesis, we extract building footprints from open-source datasets such as DSM and DTM from AHN4, and aerial ortho MSI imagery. Using elevation and NDVI thresholds, we identify and isolate likely building structures in a step-by-step process. Our assumption is that buildings generally have lower NDVI values and are elevated above the surrounding ground, allowing for effective differentiation from vegetation and other features. Refer to Figure 3.4 for theoretical pipeline and Figure 3.5 for illustration. The detailed steps used to extract the building footprints are as follows:

1. **Binary elevation map**: First, we create a binary raster map (0.5m resolution) from the DSM, highlighting pixels that are higher than 2m above the average ground elevation (mean DTM elevation). Next, we convert the binary map into polygons, assuming that all DSM pixels exceeding the 2m threshold represent buildings, trees, poles, or noise.

2. **Smoothing and noise removal**: To remove tree and noise polygons, the polygons are eroded and then dilated by 1m. This process may still leave polygons representing trees clustered in areas larger than $1m^2$.

3. **Mean NDVI and buildings extraction**: We generate NDVI raster maps using the equation 2.2 in Chapter 2.3.1. The extracted polygons are then overlaid onto the NDVI layer, and the mean NDVI values are calculated by averaging the NDVI pixels within each polygon. Polygons with an NDVI value below 0.3 are classified as buildings, since buildings typically have low NDVI while trees have high NDVI.

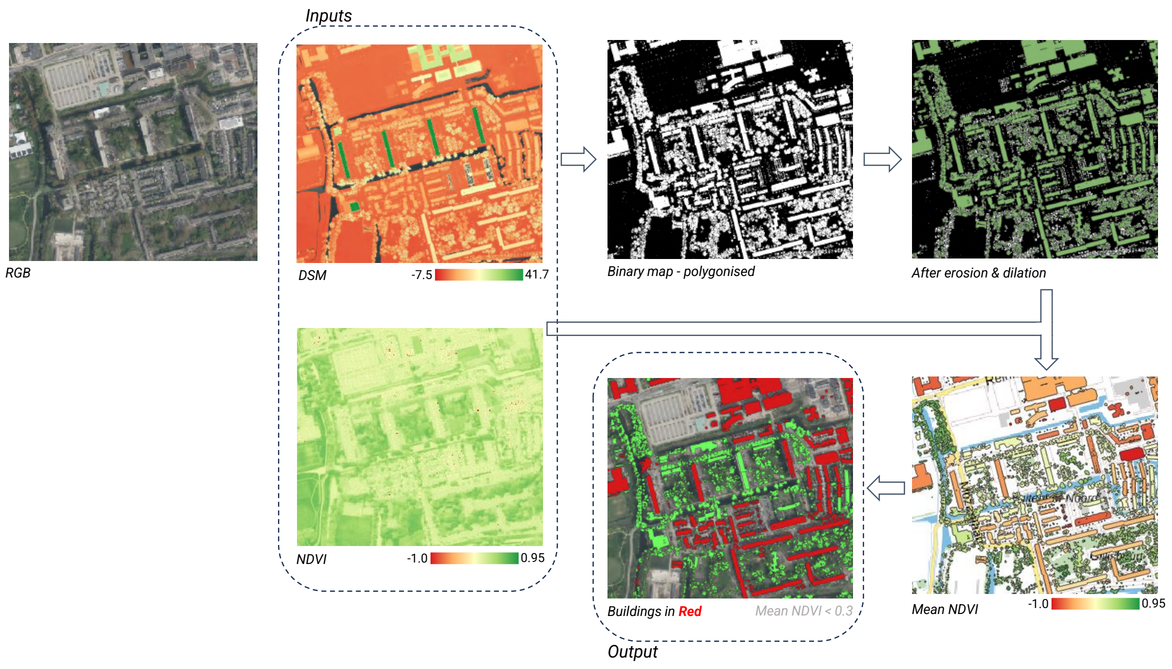
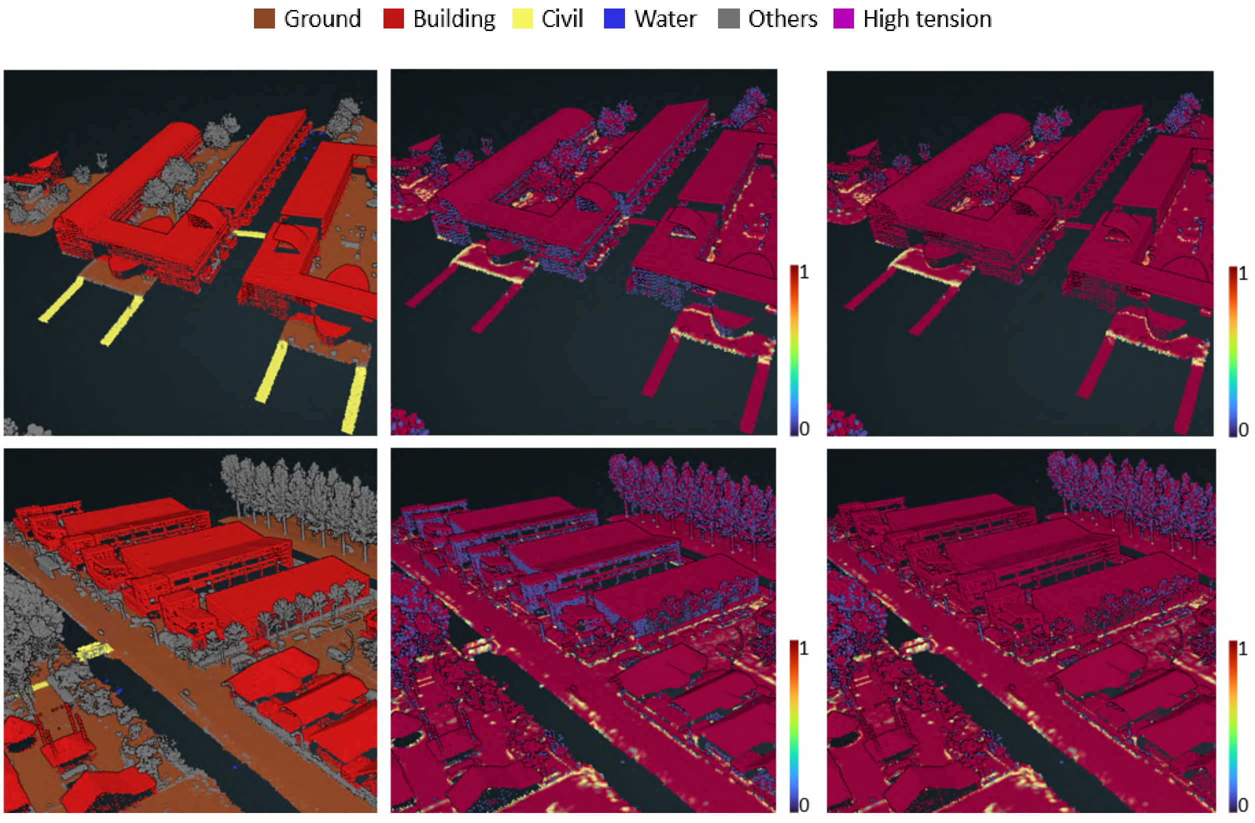


Figure 3.5: Buildings footprint extraction using 2D datasets

Finally, for the points labeled as *building* in the point cloud and falls inside a building polygon, their confidence is increased to one, and *Ground* labeled points confidence is reduced to zero. Figure 3.6 compares primary and refined confidence scores.



(a) Annotated point cloud (b) Primary confidence (c) Refined confidence

Figure 3.6: Confidence scores — *Primary and Refined*

After refining the confidences of the points, they are then segregated into two categories based on their confidence scores: *under-confident* and *over-confident* points, using a custom threshold, t_1.

It is also possible to enhance *Building* point confidences using 3rd party building footprints resources such as 2D BAG. However, experiments suggest that our extraction method achieves on-par performance as 2D BAG. In some areas we are able to detect building footprints with higher precision. Please refer to Chapter 6.1.3 for detailed discussions.

3.2 Network training

The next step in the workflow is training the deep learning model. Here we want the *over-confident* point samples to guide the learning process, and not let *under-confident* samples to participate and pollute training of the model. However, the approach has a challenge, that is, less data participates in the training process, as we mask under-confident points. Our method, *online deep learning* strategy, address this by iteratively refining the confidences and labels of *under-confident* points and makes them participate in the training. The step by step process is as follows (Figure 3.1):

1. **Network training with confident samples:** We start by selecting a state-of-the-art deep learning model for point cloud segmentation, which is known for its flexibility and adaptability. The model serves as the *backbone network*, which will be trained and tested. For the thesis, **KPConv** model proposed by Thomas et al. [2019] is chosen. The model is first trained only on *over-confident* point samples for a fixed number of epochs e, in order to learn good feature representations.
2. **Model predictions:** After the initial training, the model is then used to make predictions on all the points in the cloud, including both over and under confident points.
3. **Psuedo-label selection:** The model's predictions yield probabilities (via softmax outputs, detailed in the appendix). We select the most confident predictions using a second threshold, known as the *online confidence threshold* t_2. These strong predictions replace the original ground truth labels, creating new labels referred to as *pseudo-labels*. This approach enables the deep learning model to dynamically adjust both the confidence values and existing labels of the training points from which it learns, enhancing its ability to refine and update its understanding of the data during the training process.
4. **Iterative training:** The model is retrained iteratively, incorporating these pseudo-labels in each iteration, thereby refining its performance.
5. **Outputs:** The process yields two main outcomes. First, a cleaned point cloud dataset with reduced or no under-confident points, as they have been relabeled using the model's confident predictions. Second, a robustly trained model capable of accurate semantic segmentation of point clouds in unseen urban scenes.

Our online learning strategy shares some similarities with the self-training technique used by Li et al. [2021]. In self-training, there is limited training data, which is augmented using a GAN. This involves two networks—a segmentation network that makes predictions and a discriminator network that evaluates them—working together. However, this method can be computationally demanding. In contrast, our online learning method uses only a single segmentation network by incorporating prior geospatial knowledge during the preprocessing phase to provide confidence scores, making our model more lightweight.

3.2.1 Backbone network - KPConv

For this thesis project, we selected the KPConv model [Thomas et al., 2019], a point-based convolutional network, as our backbone network. Details about this model are covered in Chapter 2.6.2. Although there are other state-of-the-art transformer-based models, such as Point Transformer and Stratified Transformer [Zhao et al., 2020b; Lai et al., 2022], we opted for KPConv because it strikes a good balance between performance and computational cost. In principle, any deep learning network could serve as the backbone since our primary interest is exploring how online learning can improve the model, rather than determining the best deep learning architecture.

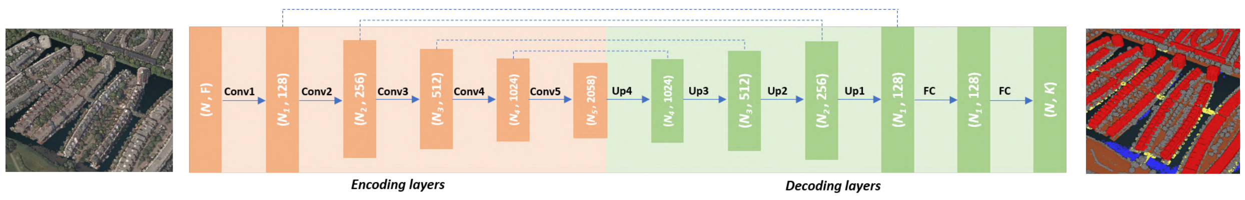


Figure 3.7: Backbone KPConv network used for the project

The following network parameters are implemented:

1. **Input**: $N \times (3 + F)$, where F is input feature dimension. Multiple feature combinations are implemented for the thesis:

F	features
2	elevation, intensity
3	elevation, intensity, NIR
5	elevation, intensity, red, green, blue

Table 3.1: Features used for deep learning model

2. **Number of classes**: 6. They are *other*, *ground*, *building*, *water*, *high-tension*, and *civil structures*, with label values of *1*, *2*, *6*, *9*, *14*, and *26*.
3. **Encoder**: 5 layers, and each layer has 3 convolutional blocks, with first one being strided (for downsampling) except for first layer. Each block consists of KPConv, batch normalization, and leaky ReLu activation.
4. **Decoder**: 4 layers, with each layer consisting nearest neighbor upsampling, and unary convolution.
5. **Output**: $N \times K$, where K is the number of classes. In this thesis, $K = 6$. Output values represents probability predictions of points belonging to a particular class.

3.2.2 Class imbalance

Class imbalance is a situation where certain classes dominate others. For instance, most points in the point clouds of AHN4 belong to *ground*, *building*, *water*, and *other* categories, while the number of points belonging to *civil* or *high-tension* are relatively lower.

The points that participated in training across classes in training datasets is shown in Table 3.2.

Others	Ground	Building	Water	High-tension	Civils	Total
102,865,725	215,135,387	66,534,958	16,900,460	47,943	1,126,217	402,610,690

Table 3.2: Training dataset point cloud distribution

Due to the fact that there is less data available to learn from minority classes, the model faces obstacles in learning useful features, and this is because loss function is overly exposed to only a few classes [Griffiths and Boehm, 2019]. Consequentially, the segmentation quality of minority-class objects would also be relatively low.

A common technique in deep learning (DL) to address class imbalance is by adjusting the loss function to give more weight to underrepresented classes. One effective method is *focal loss*, introduced by Lin et al., 2017. *Focal loss* assigns weights to the loss based on class frequency, so classes that appear less often receive higher weights. These weights ensure that the loss function is not overwhelmed by dominant classes, ensuring better performance on rare classes.

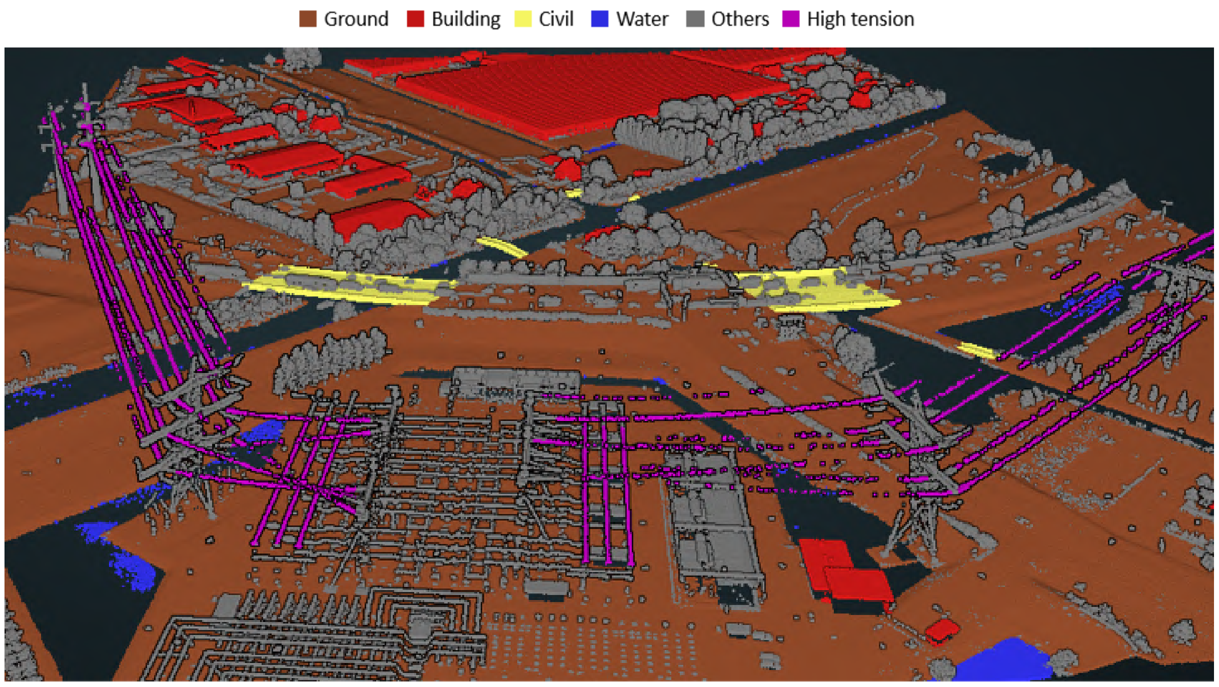


Figure 3.8: A sample training tile showing the scarcity of high tension and civil points

In our study, given prior knowledge of class frequencies, we used a weighted cross-entropy loss. The weights are calculated based on the "inverse proportion of class frequencies", as shown in the equation below:

$$p_c = \frac{n_c}{N} \tag{3.1}$$

where c is a given class, n_c is the number of points in class c, N is the total number of points in training and p_c is the proportion of points in class c. Final class weights are calculated using the following equation (Eq. 3.2):

$$w_c = \sqrt[3]{\frac{p_{max}}{p}} \tag{3.2}$$

where p_{max} is the proportion of the majority class. The cubic root scaling ensures a balanced yet nuanced adjustment, amplifying the importance of the underrepresented classes without overly penalizing the overrepreseneted ones. These weights are integrated into the cross-entropy loss function (Eq. 3.3):

$$L_{\text{cross-entropy}}(\hat{y}, y) = -\frac{1}{N}\sum_{j=1}^{N}\sum_{c=1}^{M} w_c y_{c,j} \ln(\hat{y}_{c,j}) \tag{3.3}$$

where N is the total number of labeled points, and M is the number of classes, $y_{c,j}$ is a binary indicator (0 or 1) of a point belonging to a certain class c, and $\hat{y}_{c,j}$ is the model's prediction probability of a point belonging to a certain class c.

3.3 Evaluation metrics

The predictions are compared with the original labels, assuming them as ground truth, using the following metrics:

1. **Per-class IoU**: Intersection over Union is a metric to measure the overlap between predicted and ground truth labels of a specific class in segmentation tasks.

$$IoU_c = \frac{AreaofOverlap}{AreaofUnion} = \frac{TP}{TP + FP + FN} \tag{3.4}$$

   where *TP* is *true positives*, *FP* is *false positives*, *TN* is *true negatives*, *FN* is *false negatives*.

2. **Per-class accuracy**: is the proportion of correctly classified samples for a specific class out of all actual samples of that class.

$$accuracy_c = \frac{TP}{TP + FN} \tag{3.5}$$

3. **Mean IoU and mean accuracy**: It is the average of per-class IoU scores, and average of per-class accuracy scores across all classes.

$$\text{mIoU} = \frac{1}{c}\sum_{i=1}^{c} IoU_c \tag{3.6}$$

$$\text{mAcc} = \frac{1}{c}\sum_{i=1}^{c} IoU_c \tag{3.7}$$

4. **Overall accuracy**: is the fraction of all correct predictions in the whole dataset.

$$OverallAccuracyOA = \frac{\text{Correct precdictions}}{\text{Total number of predictions}} \tag{3.8}$$

The online deep learning model predictions are evaluated using the above formulas. However, during the training process, the model updates the training samples which are considered as ground truth labels. The changes in the training data qualitatively evaluated and understood by visual inspection, but not through any quantitative measurements.

4 Implementation

In this chapter, we discuss the practical aspects of implementing the proposed deep learning framework. This includes a description of the dataset, hyperparameter settings, and the hardware and libraries used.

4.1 Data

The primary dataset used in this thesis is Actueel Hoogtebestand Nederland (AHN), specifically the AHN4 version. Detailed information about AHN4 is provided in Chapter 2.2. AHN4 data is publicly available, which can be accessed from official website here: https://www.ahn.nl/ahn-viewer.

The AHN dataset from the original source includes standard point cloud attributes such as intensity, return number, number of returns, classification labels, GPS time, etc. However, it does not include RGB colors or NIR (Near-Infrared) reflectance values, meaning the original AHN4 dataset consists of uncolored point clouds.

To address this, GeoTiles has integrated RGB and NIR data into the point cloud using publicly available aerial imagery. The GeoTiles colored point cloud, which includes both RGB and NIR values, can be accessed here: https://geotiles.citg.tudelft.nl/. Since our project uses RGB and NIR values as input features for model training, we utilized the GeoTiles point cloud data for this purpose.

GeoTiles provides a finer tiling of the AHN dataset, where each tile covers 1×1.25 Km and includes a 20-meter overlap with neighboring tiles. Due to hardware limitations when handling large datasets, each tile is further subdivided into four mini tiles, each measuring 0.25×0.3125 Km, with a 10-meter overlap with neighboring mini tiles. The dataset is split approximately 85% for training and 15% for validation, utilizing 52 mini tiles for training and 8 mini tiles for validation. The number of points per class participated in the training process is described in the Table 3.2, and validation dataset point distribution in the Table 4.1.

Others	Ground	Building	Water	High-tension	Civils	Total
17,605,936	37,244,632	5,804,124	10,381,515	1,461	33,819	71,071,487

Table 4.1: Validation dataset point cloud distribution

The point per class distribution is highly skewed. We have *high-tension* and *civil structure* points as minority classes, as they are in thousands, whereas other classes are present in millions.

4.2 Hyperparameters and deep learning training details

This section outlines the hyperparameters crucial for both the preprocessing and online deep learning stages, as shown in Table 4.2. These hyperparameters are essential for the preprocessing of point cloud data and the online deep learning process, influencing the model's performance and ensuring the reproducibility of results.

Preprocessing		
Hyperparameter	Value	Remark
r	0.5 m	Neighborhood radius for primary confidence measurement
t_1	0.9	Confidence threshold for *under* and *over* confident segregation
Backbone network — KPConv		
Hyperparameter	Value	Remark
$first\_subsampling\_dl$	0.2 m	Voxel size for downsampling
N	300	Training epochs
$Epoch\ steps$	300	Number of batches in each epoch
lr	0.01	Learning rate
in_radius	10.2	Neighborhood selection radius for each point
$conv\_radius$	2.5	Radius of the convolution sphere
$kernel\ points$	15	Number of kernel points
KP_extent	1.2	Spatial extent of the kernel points
Online learning specific		
Hyperparameter	Value	Remark
e	150	Epoch at which online learning begins
t_2	0.99	Online confidence threshold

Table 4.2: Preprocessing and Network training hyperparameters

4.3 Hardware and libraries

In this section, we outline the computational resources, python libraries, and visualization tools utilized in the project, highlighting the hardware infrastructure for preprocessing and deep learning, along with the key software libraries that facilitated data handling and processing.

1. **Hardware**: The hardware setup includes CPU-based preprocessing for efficiency and GPU-powered deep learning to handle computationally intensive tasks.

   a) ***Preprocessing***: The preprocessing pipeline was fully executed on CPUs, with no GPU resources utilized. To improve efficiency, various workflow components were parallelized using multi-threading techniques.

   b) ***Deep learning***: For model training and evaluation, we utilized the DelftBlue HPC cluster from the Delft High-Performance Computing Centre (DHPC) [Delft High Performance Computing Centre , DHPC]. The computations were powered by NVIDIA Tesla V100 GPUs (32 GB memory), optimizing the deep learning workload.

2. **Libraries**: A variety of Python libraries were used to manage, process, and analyze geospatial and point cloud data, alongside tools for deep learning and scientific computation.

   a) **PyTorch** [Paszke et al., 2019]: The deep learning framework is built using the PyTorch framework.

   b) **laspy**: This Python library handles operations such as reading, writing, and manipulating LAS/LAZ files. It supports filtering and processing point cloud data.

   c) **scipy**: A core Python library for scientific computation. In this project, scipy was used to identify nearest neighbors through kd-tree structures.

d) **rioxarray**: A python library which is an extension of xarray library. It is used for handling geospatial raster data (GeoTiff).

e) **GeoPandas**: A python library used for geospatial vector data.

3. **Visualization tools**:

a) **Potree**: Potree is an open-source web-based renderer for visualizing large 3D point cloud datasets efficiently in web browsers. It's accompanied by Potree Converter, a tool that transforms point cloud data into a specialized, web-friendly format with a hierarchical structure [Schutz et al., 2020]. This combination enables smooth rendering and progressive loading of massive datasets without requiring high-end hardware or software installations. Additionally, Potree offers extensive customizability through JavaScript and HTML web development kits, allowing users to tailor the visualization experience to their specific needs and integrate it into web applications.

For this thesis Potree has been utilized at every step of the project. Starting from visualizing the confidences in preprocessing to finally viewing the prediction results of our deep learning models.

b) **QGIS**: QGIS (Quantum GIS) is an open-source GIS software used for viewing, analyzing, and editing geospatial data. It supports a wide range of vector, raster, and database formats, making it a powerful tool for mapping and spatial analysis. In this thesis project we have used it for 2D visualizations of satellite MSI, intermediary outputs of NDVI and preprocessing steps, especially 'Refining confidences'.

5 Results

The following chapter presents the experimental results for this thesis project. First, Section 5.1 examines segmentation results and compares the performance of deep learning models trained on point clouds using base features—elevation and intensity. Subsequently, the models' segmentation results will be evaluated and compared when additional features, such as RGB and NIR, are incorporated. Section 5.3 discusses online updates to the training data to provide a clearer understanding of their impact.

In the comparison tables in the following sections, the backbone network-KPconv, trained on just the overconfident samples, without online learning implementation is referred to as ***'Baseline'*** model. Backbone network-KPConv, trained with online learning strategy using the primary confidence scores, is referred to as ***'+Online'*** model. Because online learning is a strategy that is applied on top of baseline model. For reference, *Baseline* model is left-side model implementation, and *+Online* model is right-side model implementation in Figure 3.1. Due to high variations of network training outputs, for each setting, we experimented with three training runs and reported the averaged scores to achieve a fair comparison.

5.1 Results with base features — Elevation, Intensity

Intensity is one of the important features of LiDAR point cloud data. It signifies the return strength of the laser pulse that generated the point, and its value is based on the reflectivity of the object struck by the laser pulse. For point cloud classification, intensity values are used as an aid in feature detection and extraction [ESRI, 2023]. Since intensity is collected during the data acquisition stage—unlike RGB values, which are fused later to the point cloud—it naturally serves as a foundational feature for training models.

Similarly, *elevation*, another critical base feature, provides essential contextual information about the positioning of points within the 3D space. This information aids the model in understanding variations in ground levels, structures, and landscape features, offering spatial context necessary for segmentation tasks.

Both intensity and elevation are chosen as base features because they complement each other by capturing different aspects of the environment—intensity reflects the material properties of surfaces, while elevation provides crucial spatial information. The below Table 5.1 shows the performance of models when trained with base features, and Figure 5.1 visualizes the predictions of models.

Overall, our *+Online* model has a slight edge over the baseline model, outperforming it in two of the three key overall metrics. While the baseline achieves a slightly higher mAcc—79.6% compared to the online model's—79.4%, the online strategy surpasses the baseline in both overall accuracy (OA) and mean Intersection over Union (mIoU), scoring 95.1% in OA versus the baseline's 94.8%, and 65.0% in mIoU compared to 63.8% for the baseline. These results suggest that online learning enhances the model's ability to learn more effective features, leading to overall better performance. This improvement could be attributed to the fact that online learning exposes the network to a larger number of samples, that are highly likely to have good quality which are refined and updated during the training iterations.

A notable improvement in accuaracy is seen in the *Others* class (89.8% to 90.8%), with slight gains in *Building* (78.1% to 79.4%) and *High tension* (32.2% to 33.2%), indicating benefits from the online strategy. The *Water* class maintains a high accuracy of 99.2% across both approaches, showing consistent performance. However, the *Civil* class experiences a decrease in accuracy with online learning (80.0% to 75.3%), suggesting challenges for this class. Despite an accuracy above 75% for both approaches, IoU for the *Civil* class remains very low, at less than 6%. Importantly, online learning excels in IoU performance across all classes, with notable gains in *Building* (73.5% to 75.4%) and *High tension* (27.4% to 30.4%).

Model features: *elevation, intensity*								
	Per class accuracies						**mAcc**	**OA**
	Others	Ground	Building	Water	High tension	Civil		
Baseline	89.8	**98.6**	78.1	**99.2**	32.2	**80.0**	**79.6**	94.8
+Online	**90.8**	**98.6**	**79.4**	**99.2**	**33.2**	75.3	79.4	**95.1**
	Per class IoUs						**mIoU**	
	Others	Ground	Building	Water	High tension	Civil		
Baseline	**86.6**	**94.8**	73.5	98.1	27.4	2.6	63.8	
+Online	85.4	**94.8**	**75.4**	**98.4**	**30.4**	**5.7**	**65.0**	

Table 5.1: Baseline and online learning performance comparison with accuracies and IoUs, when trained with base features—elevation, intensity. (All values represent the average of three experiments, ensuring fair comparison)

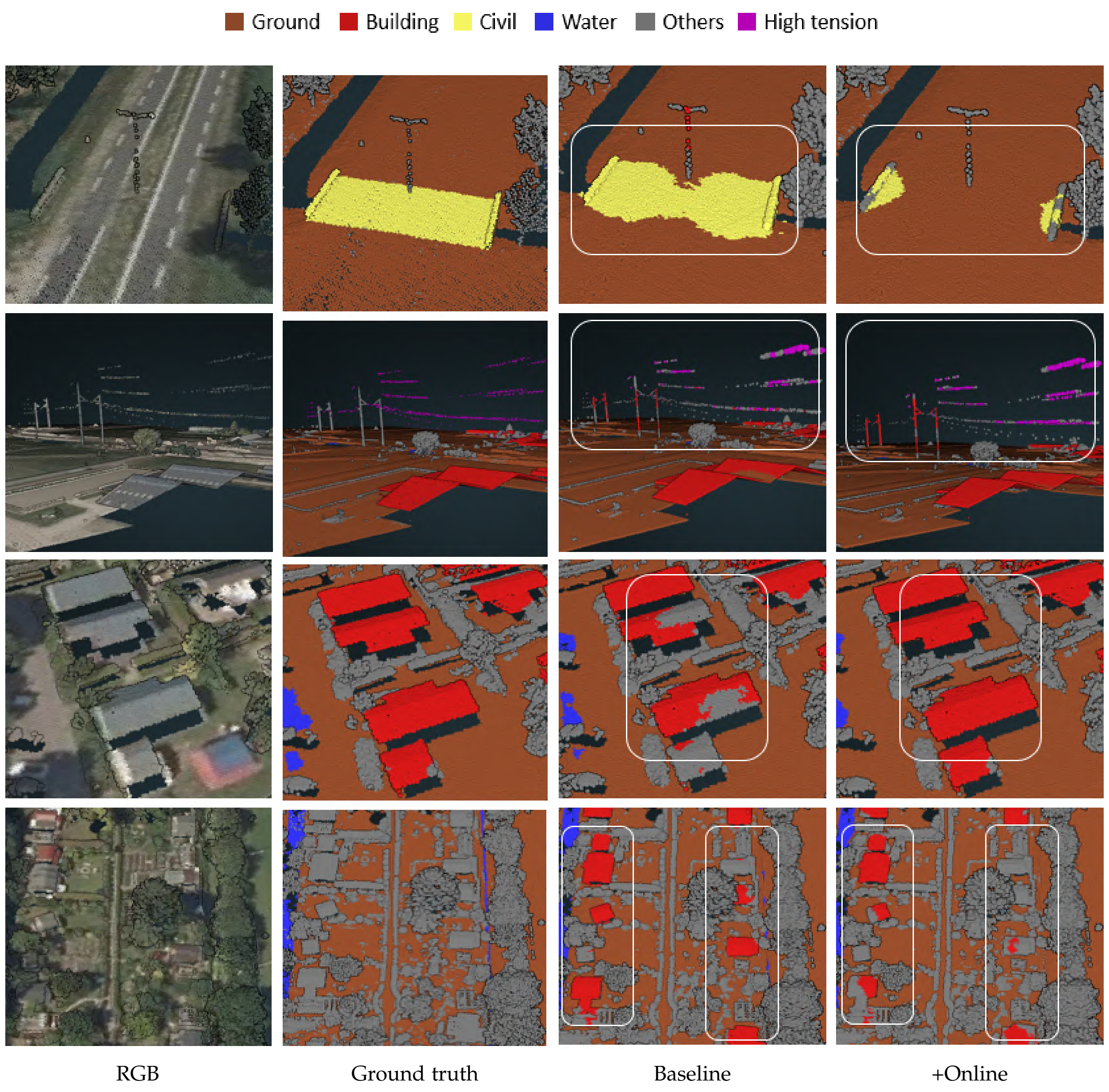


Figure 5.1: Deep learning models comparison when trained on base features — *Elevation* and *Intensity*

Given the raw attributes of the point cloud data—intensity and elevation—as features, these results highlight the online learning strategy's effectiveness in enhancing the model's overall performance and generalization ability across classes.

5.2 Results with additional features

In this section, we incorporate additional features into the deep learning models, specifically adding components such as RGB and NIR, which can be considered supplementary to the base features of elevation and intensity. These features, as discussed in Section 2.1.2, represent secondary components of the point cloud. Our aim is to evaluate how the inclusion of these additional features affects the model's performance compared to when it is trained solely with the base features.

To read the tables, the arrows indicate the direction of change in the model's performance when additional features (such as NIR or RGB) are included, compared to the base features (elevation and intensity). The numbers in brackets show the difference in accuracy or IoU relative to the base model. An upward arrow (↑) represents an improvement, while a downward arrow (↓) signifies a decline in performance.

5.2.1 Results with additional NIR feature

In Table 5.2, the performance of the *Baseline* model and the *+Online* model is compared, when they are trained with elevation, intensity and NIR features.

Model features: *elevation, intensity, NIR*								
	Per class accuracies						**mAcc**	**OA**
	Others	Ground	Building	Water	High tension	Civil		
Baseline	**91** ↑ (1.1)	98.4 ↓ (-0.3)	**73.5** ↓ (-4.6)	99.2 (0)	**44.8** ↑ (12.6)	**79.0** ↓ (-1.0)	**81** ↑ (1.3)	94.6 ↓ (-0.2)
+Online	90.8 (0)	**98.6** (0)	72.9 ↓ (-6.5)	**99.3** ↑ (0.1)	30.9 ↓ (-2.3)	77.6 ↑ (2.3)	78.3 ↓ (-1.1)	**94.7** ↓ (-0.5)
	Per class IoUs						**mIoU**	
	Others	Ground	Building	Water	High tension	Civil		
Baseline	86.5 ↓ (-0.1)	**95.1** ↑ (0.3)	69.1 ↓ (-4.3)	**98.4** ↑ (0.3)	**38.3** ↑ (10.9)	**2.5** ↓ (-0.1)	**65** ↑ (1.2)	
+Online	**87.0** ↑ (1.6)	**95.1** ↑ (0.2)	**69.6** ↓ (-5.9)	97.9 ↓ (-0.5)	28.8 ↓ (-1.7)	2.4 ↓ (-3.3)	63.4 ↓ (-1.6)	

Table 5.2: Generic and online learning performance comparison with accuracies and IoUs, when trained with three features—elevation, intensity, NIR. (All values represent the average of three experiments, ensuring fair comparison)

The addition of the NIR feature to the base features (elevation and intensity), led to mixed performance outcomes across different metrics and classes. When comparing the overall metrics, the addition of the NIR feature led to a decrease in overall accuracy (OA) for both the *baseline* and *+online* models, although the *+online* model maintained a slightly better OA (94.7%) compared to the baseline (94.6%). The *Baseline* model, however, showed improvements in both mean accuracy (mAcc) and mean IoU (mIoU), outperforming the *+Online* model on these metrics. Specifically, the baseline achieved an mAcc of 81.0% and an mIoU of 65.0%, while the online model had an mAcc of 73.9% and an mIoU of 63.4%. Considering these three overall metrics, the *Baseline* model emerged as the best performer when trained with three features (elevation, intensity, NIR).

Looking at class specific performance, such as *Civil*, *High tension*, and *Ground*, showed varying responses to the addition of NIR. The *Civil* class, in general, experienced a minor drop in both accuracy and IoU, with the baseline performing better in both metrics (accuracy: 77.6%, IoU: 2.4) compared to the online model (accuracy: 77.6%, IoU: 1.7). For the *High tension* class, the baseline saw a notable improvement in accuracy (44.8%), while the online model remained stable but lower (30.9%). In the *Ground* class, both the baseline and online models had similar performance, with minimal differences (accuracy of 98.4% for both). The baseline was generally more consistent in these classes, showing slightly better or comparable performance to the online model when trained with NIR.

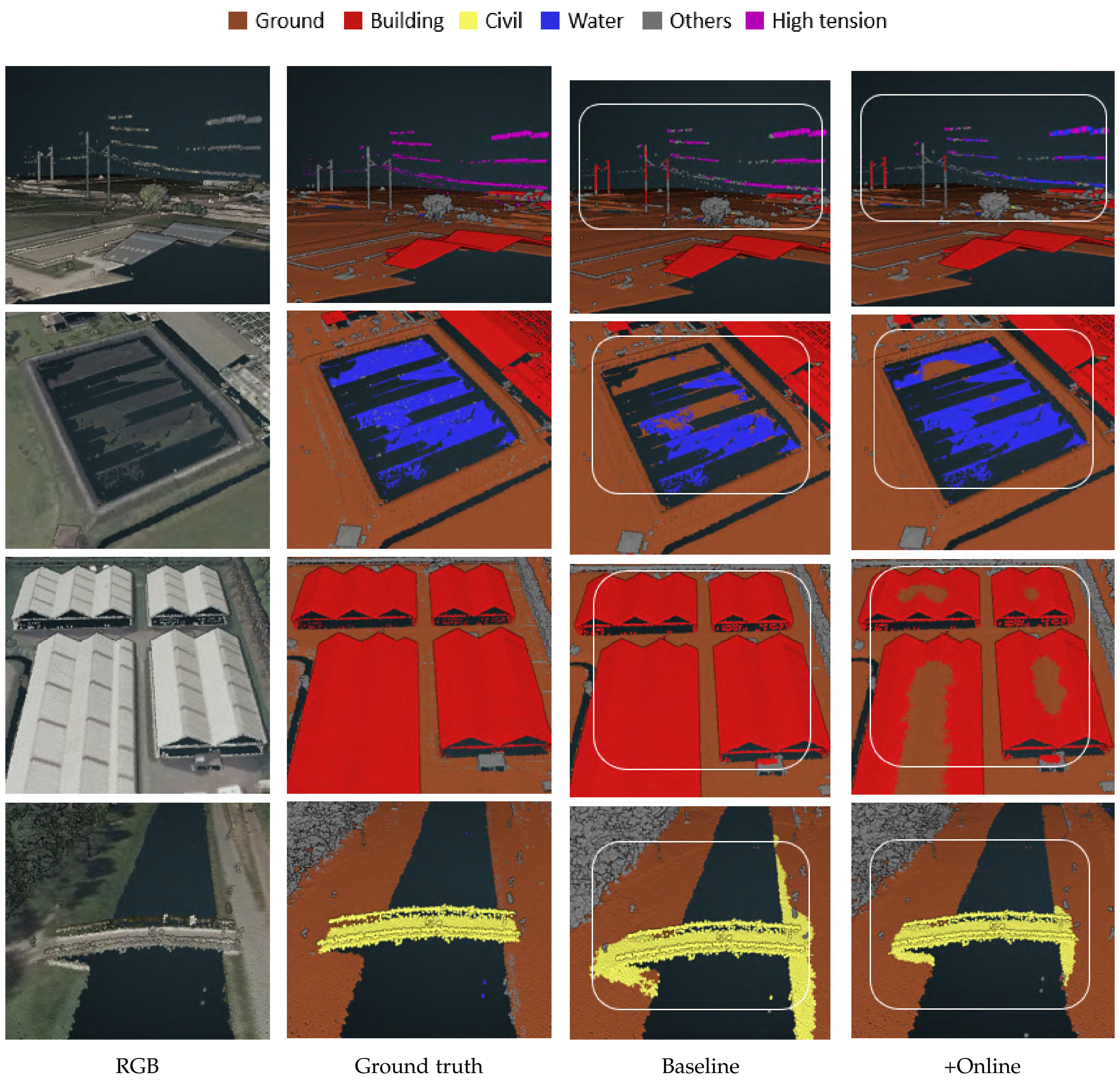


Figure 5.2: Deep learning models comparison when trained on 3 features — *Elevation*, *Intensity* and *NIR*

The mixed impact of adding NIR reflects that while some classes benefited from the additional spectral information, others experienced reduced performance. This could be due to the increased complexity of the data or differences in the class-specific contributions of NIR. For example, the *Others* class, which is largely dominated by vegetation, saw improved performance with the addition of the NIR band. This improvement is likely due to the fact that NIR is particularly effective for vegetation analysis, as it captures specific spectral properties related to plant health and density. Conversely, the *Building* class experienced significant drops in accuracy and IoU.

5.2.2 With additional RGB features

In Table 5.3, the Baseline model and the +Online model are compared when trained with five features: elevation, intensity, red, green, and blue. Figure 5.3 shows the visualizations of predictions.

Model features: *elevation, intensity, red, green, blue*								
	Per class accuracies						**mAcc**	**OA**
	Others	Ground	Building	Water	High tension	Civil		
Baseline	90.5 ↑ (0.6)	97.9 ↓ (-0.8)	**80.3** ↑ (2.2)	**99.3** ↑ (0.2)	**46** ↑ (13.8)	**84.2** ↑ (4.3)	**83** ↑ (3.4)	**94.8** (0)
+Online	**91.7** ↑ (1)	**98** ↓ (-0.6)	68.9 ↓ (-10.5)	97.2 ↓ (-2)	31.3 ↓ (-1.9)	70.4 ↓ (-4.8)	76.3 ↓ (-3.1)	93.9 ↓ (-1.2)
	Per class IoUs						**mIoU**	
	Others	Ground	Building	Water	High tension	Civil		
Baseline	**87.2** ↑ (0.6)	**94.5** ↓ (-0.3)	**75.8** ↑ (2.3)	**96.5** ↓ (-1.6)	**44.5** ↑ (17.1)	**2.7** ↑ (0.2)	**66.9** ↑ (3)	
+Online	85.3 ↓ (-0.1)	94.2 ↓ (-0.7)	66 ↓ (-9.4)	95.4 ↓ (-2.9)	25.6 ↓ (-4.9)	2.4 ↓ (-3.3)	61.5 ↓ (-3.5)	

Table 5.3: Baseline and online learning performance comparison with accuracies and IoUs, when trained with five features—elevation, intensity, Red, Green, Blue. (All values represent the average of three experiments, ensuring fair comparison)

With the addition of RGB, we observe that the *Baseline* model performed better in all the overall metrics. The *Baseline* achieved mAcc of 83% and maintained OA of 94.8%, while the *Baseline* model has a lower mAcc of 76.3% and a reduced OA of 93.9%. For mIoU, the *Baseline* outperformed the +*Online* model (66.9% vs. 61.5%). Overall, the *Baseline* model demonstrated better generalization when trained with five features (elevation, intensity, RGB).

When comparing class-specific measures to the values from models trained with just base features—elevation and intensity, the *Baseline* model consistently showed increased accuracies (except *Ground* class). The +*Online* model showed decreased accuracy across classes, with the exception of the *Others* class. With RGB features, the *Others* class improved for both baseline and online models, while the *Ground* class performance declined for both.

Even when comparing the *Baseline* and *+Online* models, the *+Online* model continued to perform poorly relative to the *Baseline*. The *Baseline* demonstrated consistently better performance across all class-specific metrics and overall metrics, reinforcing that the *+Online* strategy struggled to utilize the additional RGB features effectively, while the *Baseline* model was able to leverage them for enhanced performance.

5.3 Online learning updates

The online learning model starts by training on the highly confident samples for several epochs e. During this phase, the model learns strong feature representations from these overconfident samples. After this initial phase, the model begins refining the labels in the training data. This label refinement process happens when the model becomes highly confident about its predictions on the underconfident points. These new predictions, or pseudo-labels, gradually replace the original labels in the training dataset. By iteratively updating both the model and the training labels, the online learning strategy helps to reduce noise in the dataset and enhances the model's overall performance. The below Figure 5.5 shows a few of such improvements.

In some instances, online learning may update the training data in ways that can deteriorate its quality. As shown in 5.4, when trained with base features such as elevation and intensity, the model mistakenly updated *Ground* points to *Water*. One possible explanation for this confusion is that the intensity values of these updated points are low compared to the surrounding *Ground* points, resembling the intensity of *Water* points (since water tends to absorb most of the energy, leading to low intensity values). Additionally, these points are located in flat areas, which is another characteristic similar to water surfaces.

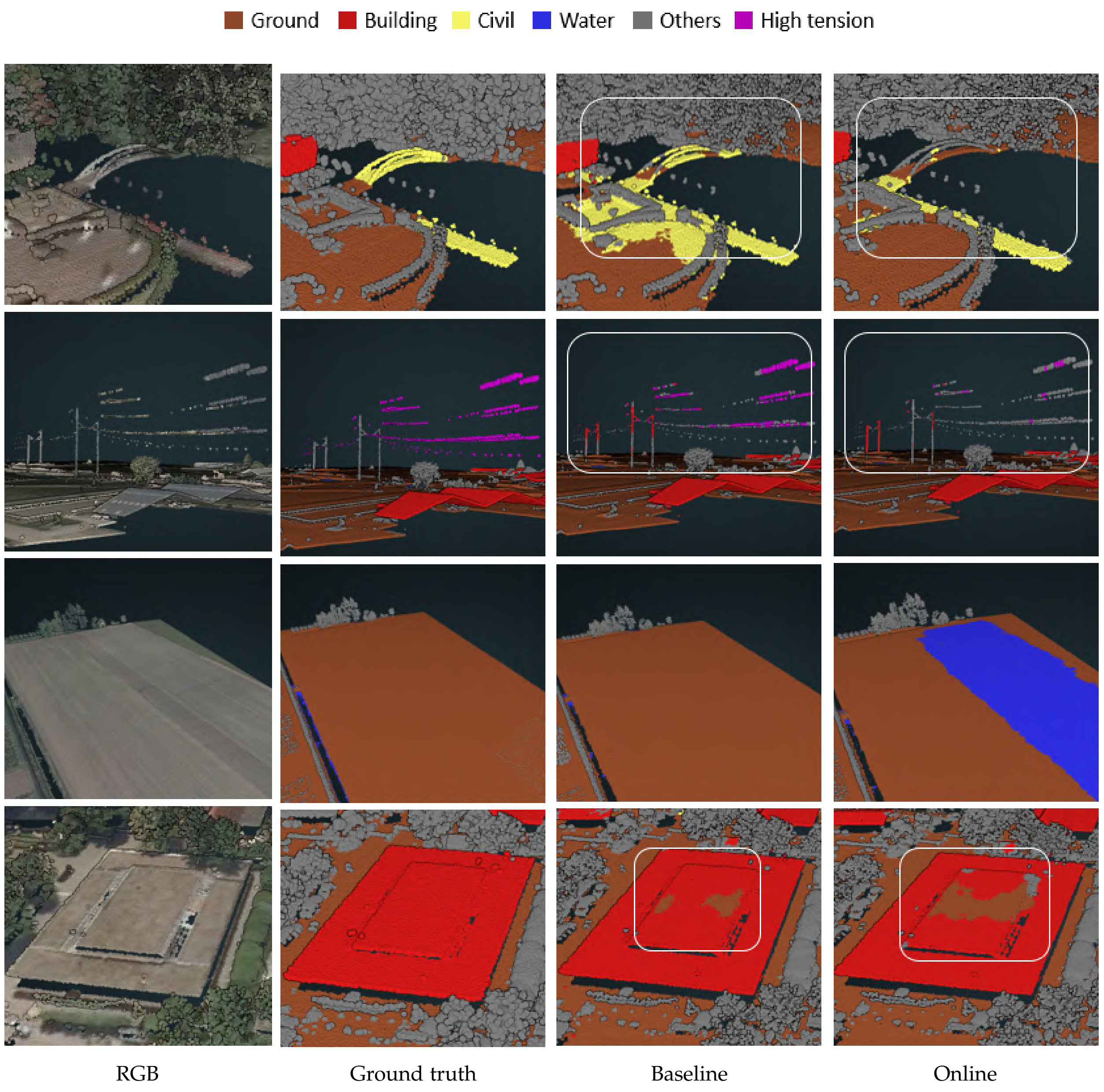


Figure 5.3: Deep learning models comparison when trained on 5 features — *Elevation*, *Intensity* and *RGB*

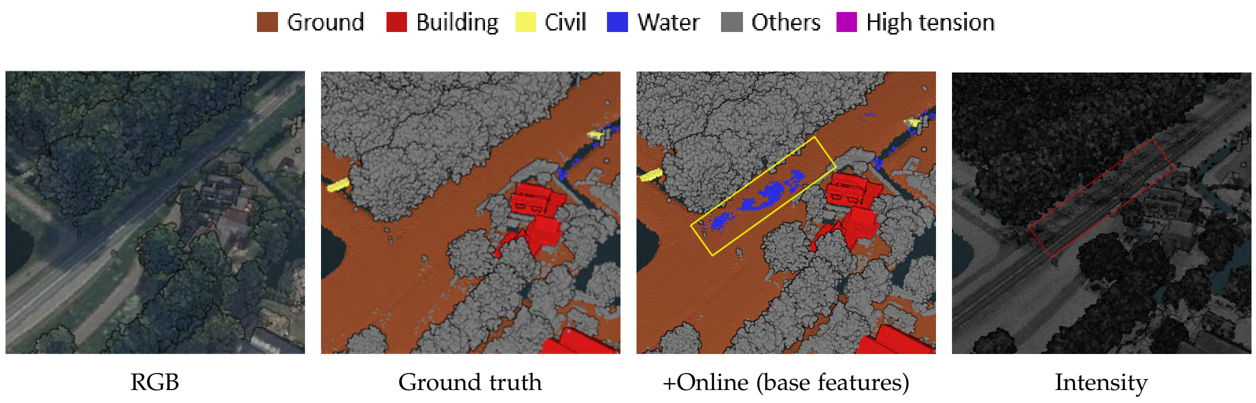


Figure 5.4: Confusion of *Ground* with *Water* because of intensity

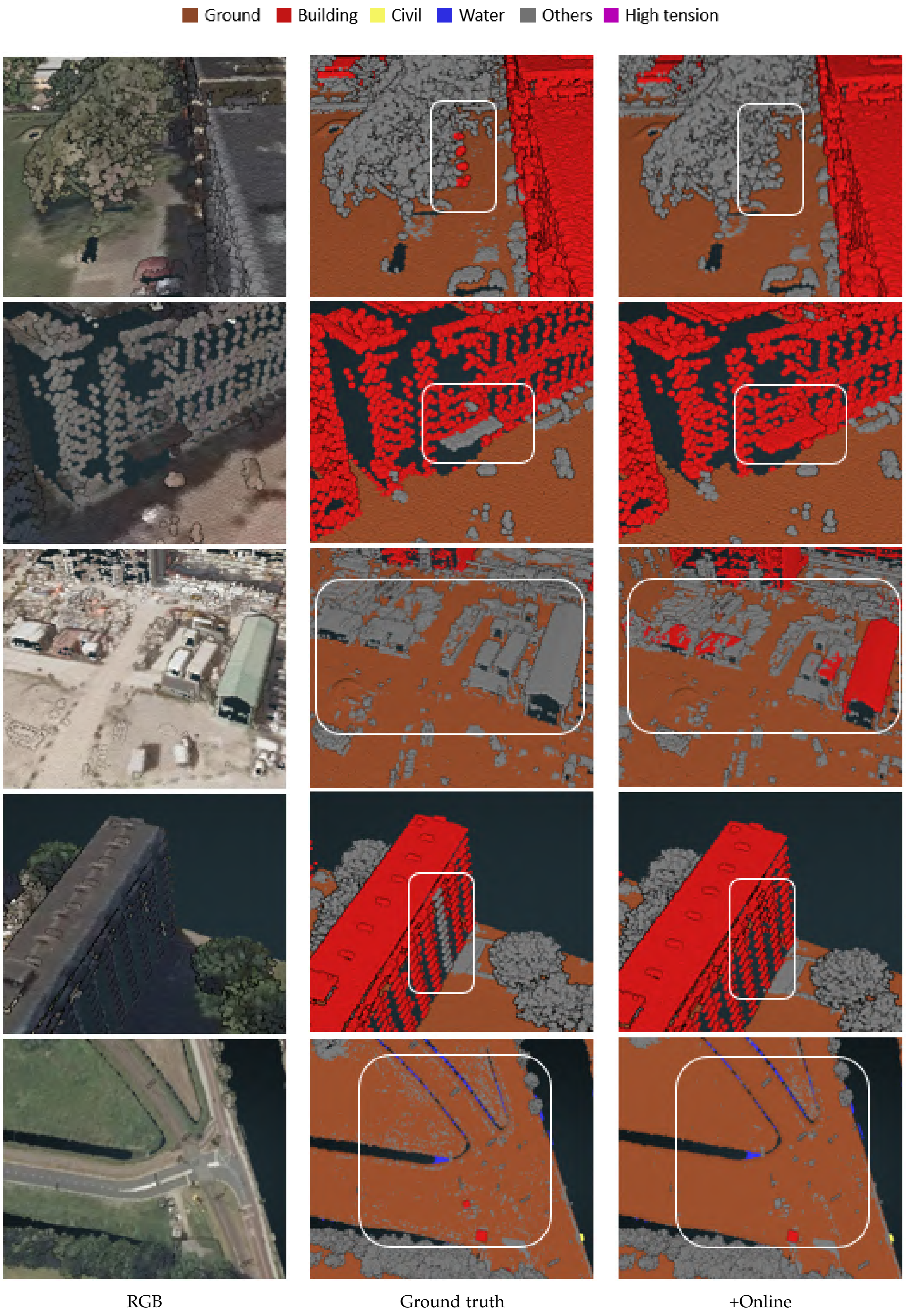


Figure 5.5: Online learning updates on training data with improvements

6 Discussion and conclusions

In this chapter, Section 6.1 provides a comprehensive discussion of the overall model performance, synthesizing key results from the previous chapter, and we examine the impact of preprocessing hyperparameters selection. This is followed by a comparison of our building footprints extracted from open-source datasets with the 2D BAG reference. Lastly, the section addresses limitations that may have constrained the deep learning models from reaching their full performance potential. Section 6.2 then provides detailed answers to each of the research sub-questions.

6.1 Discussions

6.1.1 Overall analysis

The analysis of results from the previous chapter shows that the +*Online* model with base feature—elevation and intensity, achieved the highest overall accuracy (OA) of 95.1%, demonstrating strong generalization across classes using minimal input features. However, the *Baseline* model with additional RGB features achieved the highest mean accuracy (mAcc) of 83% and mean IoU (mIoU) of 66.9%. A high mAcc indicates consistently good average prediction accuracy across all classes, while a high mIoU demonstrates better overlap between predicted areas and actual class regions, which suggests improved segmentation performance.

In conclusion, if only raw LiDAR data (intensity and elevation) is available, the +*Online* learning strategy is the most effective for segmentation. However, if additional features like NIR or RGB are available, the *Baseline* model significantly outperforms the +*Online* model, particularly for classes such as *High tension* and *Others*. Despite the improvements, it is important to note that using additional features can sometimes reduce performance for certain classes. Specifically, the *Building* class experienced a significant reduction with NIR, while the *Ground* class saw a slight reduction with RGB. Depending on specific use cases and data availability, choosing *Baseline* with or without +*Online* strategy can be determined.

6.1.2 Confidence scores — hyperparameters comparison

In the primary confidence measurement, we have two key hyperparameters: the neighborhood radius (r) and the minimum number of points in the neighborhood threshold. We assume that a radius of 0.5 m is sufficient for capturing neighborhood context. Figure 6.1 shows the histogram of the number of points within neighborhoods generated from a sample of training files. The number of points varies significantly, with higher counts observed for larger bins (e.g., 20-30 points), but the range from 1-10 points still contains a substantial number of neighborhoods, indicating that smaller neighborhoods are common. However, very small neighborhoods (<5 points) are frequent but taper off quickly and may be unreliable due to noise. Setting a threshold too high would exclude many points that provide useful local context, while a threshold too low could lead to high confidence scores for sparse, potentially noisy neighborhoods. Choosing a threshold of 5 points allows us to avoid overly sparse neighborhoods while maintaining adequate coverage for neighborhood density, effectively balancing the exclusion of single, isolated points (which could be outliers or noise) with capturing meaningful local spatial structure.

Figure 6.2 shows confidence scores for several scenes with thresholds of 5, 10, and 15. At higher thresholds (e.g., 15 and beyond), low confidence patterns appear on building roofs and the ground, failing to accurately represent neighborhood consistency due to the high number of points expected (Figures 6.2d). To address the low confidence of building wall points because of low point density with the threshold of 5 points, we employed a refining strategy (Chapter 3.1.2) for confidence scores.

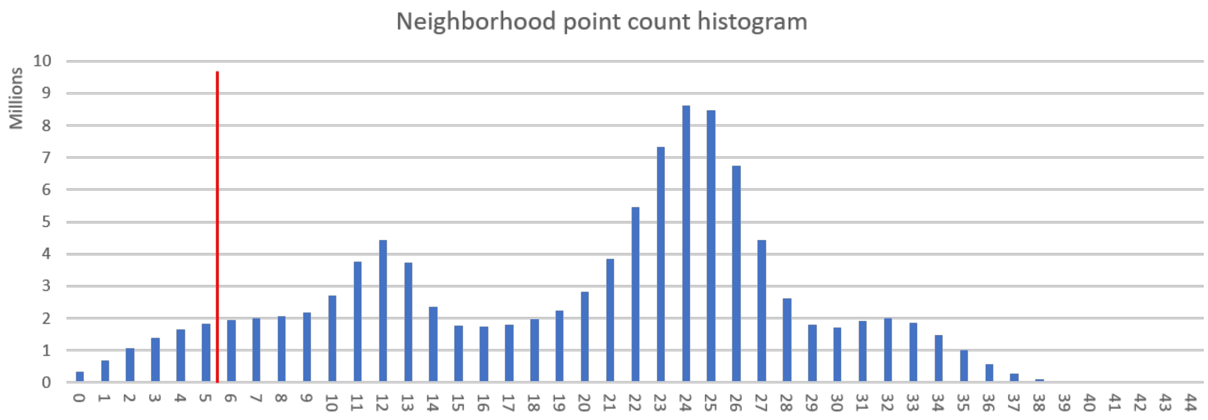


Figure 6.1: Histogram of neighborhood point counts

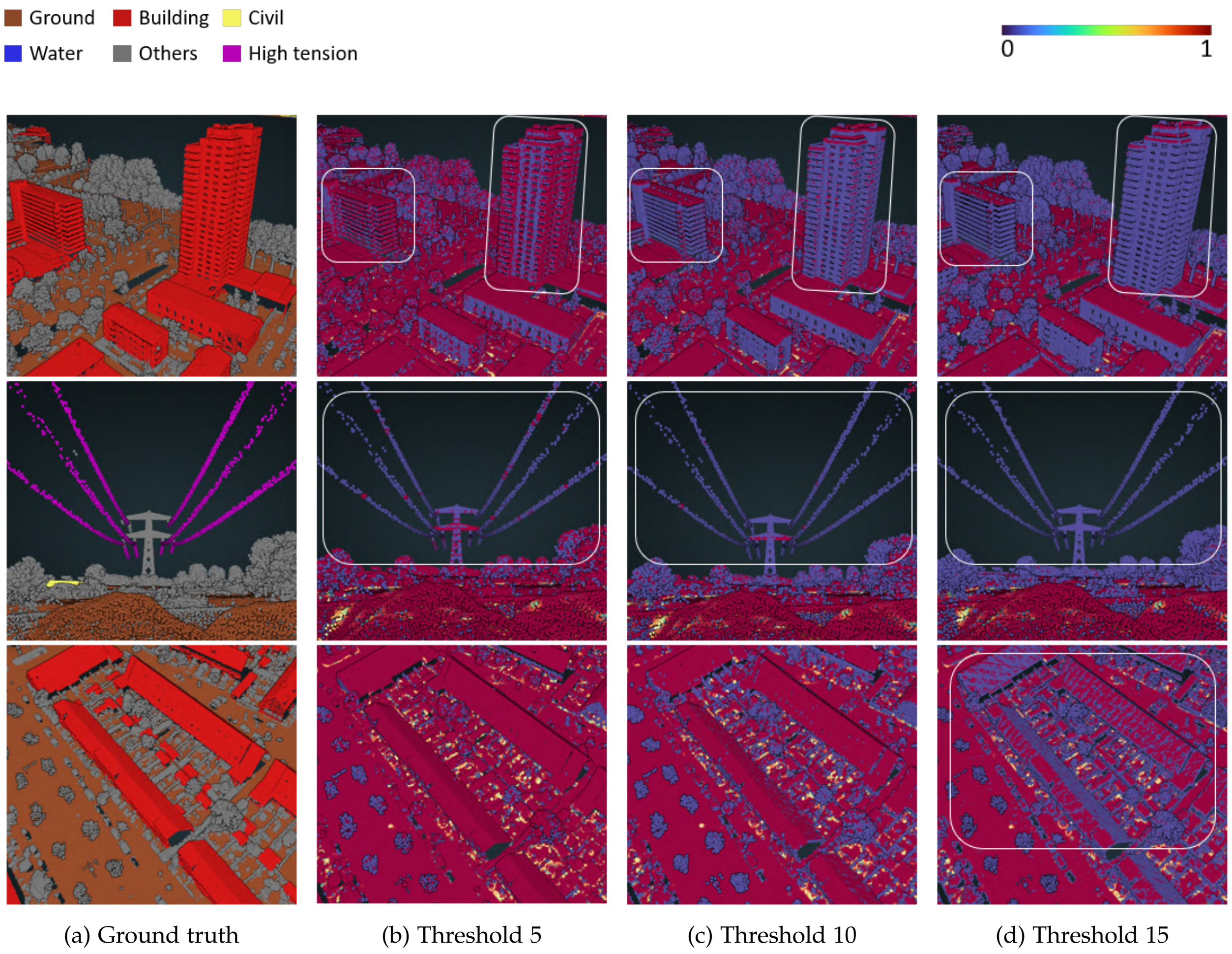
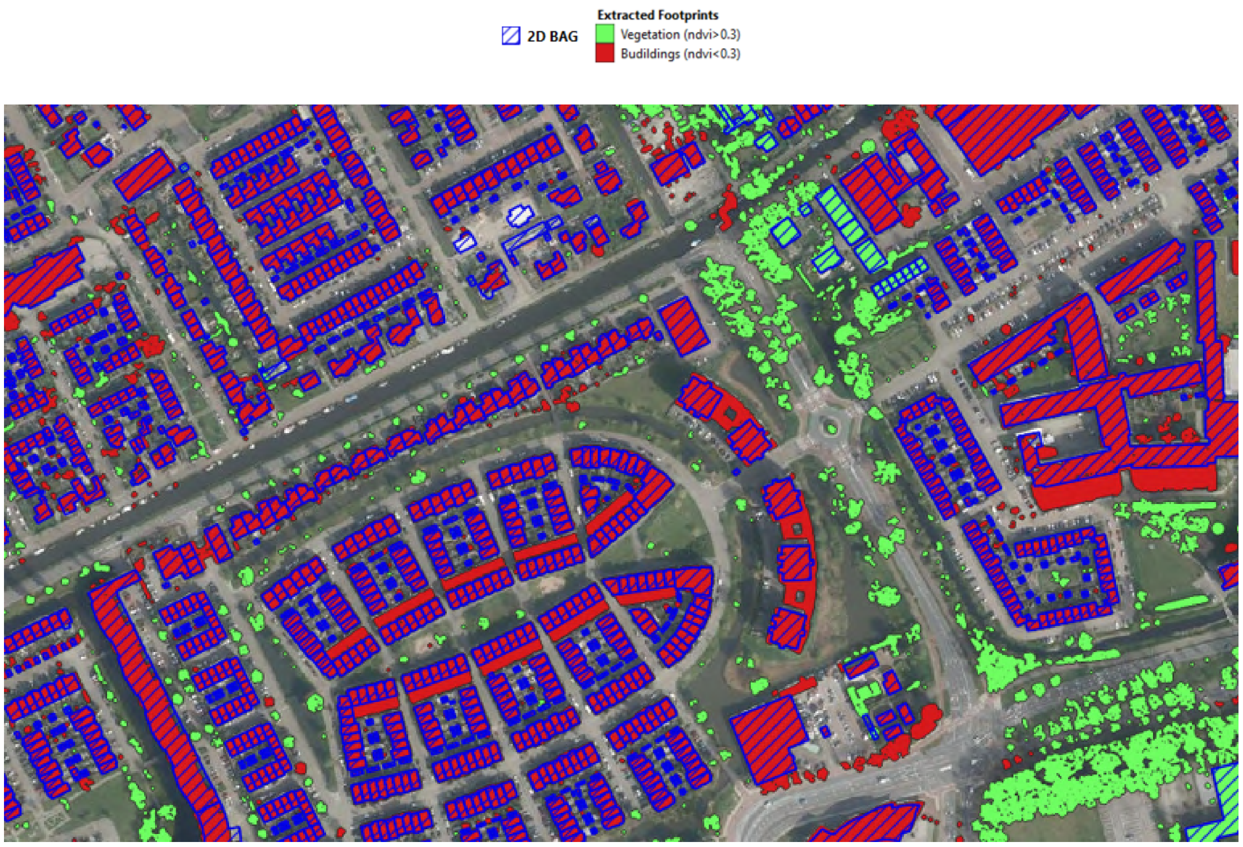


Figure 6.2: Preliminary confidence (neighborhood consistency) comparison with different thresholds

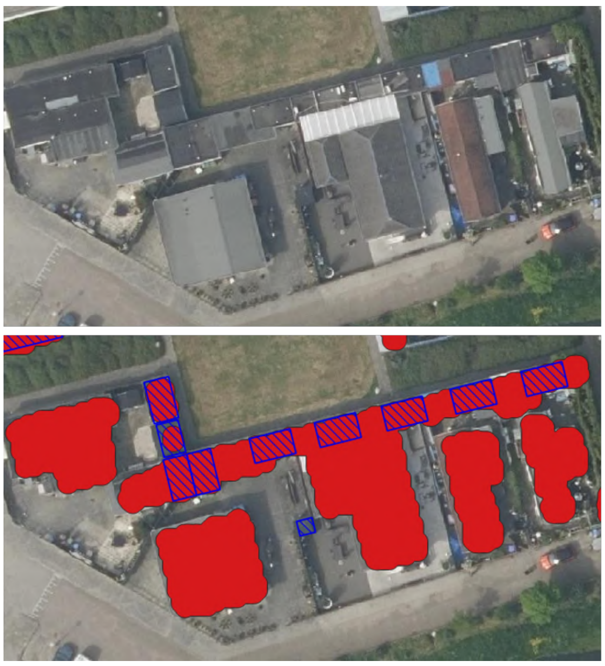
6.1.3 Building footprint comparison — Ours vs 2D BAG

To qualitatively validate the fidelity of our extracted building footprints, which were used to refine confidence scores, we overlay our building footprints with *2D BAG*[1] building polygons (Figure 6.3). Overall, there is a strong agreement between our extracted footprints and the 2D BAG polygons, suggesting that our approach of utilizing DSM and NDVI from MSI is effective. In some instances, our extracted footprints are more accurate, such as when the 2D BAG dataset has not been updated to reflect recent changes in building boundaries (Figure 6.3b). Conversely, there are situations where the 2D BAG polygons are more reliable (Figure 6.3c), particularly in areas where tree clusters are misclassified as buildings. This misclassification could be attributed to the NDVI threshold of 0.3, which may not always be sufficient to distinguish between vegetation and built structures, especially for features like greenhouses or tree canopies close to buildings. These discrepancies highlight the need for refining our approach, potentially by incorporating multi-temporal NDVI data or exploring adaptive thresholds based on local context could also further enhance the reliability of *building* footprint extraction [Huang and Zhang, 2012]. Another promising direction is to leverage recent advances in computer vision, such as the SAM2 model [Ravi et al., 2024], for extracting building footprints directly from images, which could also be utilized to refine confidence scores and improve overall accuracy.

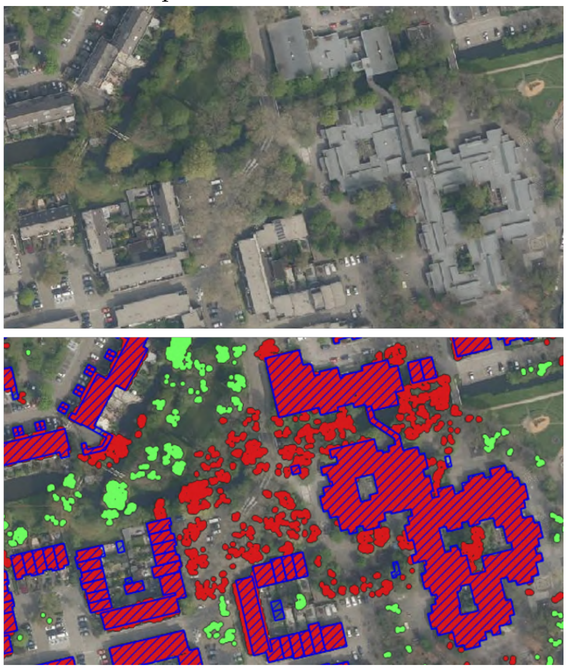
[1]The BAG dataset provides building footprint data for the whole of the Netherlands, and is available for `free download.`



(a) Overveiw of 2D BAG and our extracted footprints



(b) Our footprints better than 2D BAG



(c) Our footprints bad than 2D BAG

Figure 6.3: Comparing our building footprints with 2D BAG

6.1.4 Limitations

Although our results are promising, the performance of the deep learning models—both with and without the *Online strategy*—remains constrained due to several factors, as outlined below.

1. Model capacity

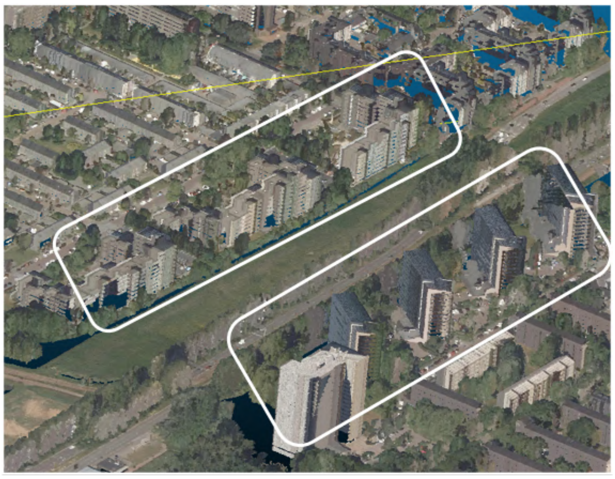
The KPConv backbone network, used in this thesis, has limitations in feature-learning capacity compared to newer transformer-based architectures. Transformer networks belong to a different family of networks that work fundamentally differently from convolutional networks. They typically incorporate *self-attention* mechanisms, which allow them to process larger geographical contexts with broader receptive fields——a capability that sets them apart from the inherently local operations of convolutional networks, limited by kernel size. For instance, recent studies have shown superior performance with transformer networks like SuperPoint [Robert et al., 2023] and Stratified Transformer [Lai et al., 2022], which leverage innovative self-attention mechanisms. Notoriously, however, self-attention is computationally intensive, requiring high computational resources compared to the more lightweight convolution-only networks like KPConv [Kappé, 2024].

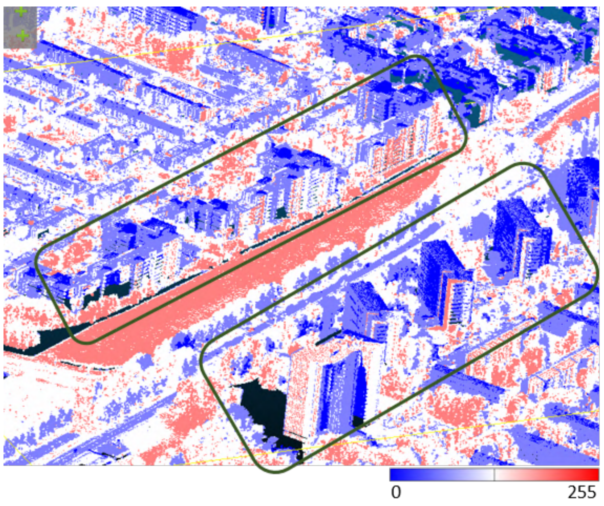
2. Quality of additional features

The raw point cloud data from AHN lacks RGB and NIR information. GeoTiles enhances this data by merging it with colored aerial MSI; however, it does not specify the source or resolution of the imagery used. Additionally, there is an inherent temporal gap between point cloud and satellite image acquisitions. Since satellite imagery is two-dimensional and typically ortho-corrected to counteract the oblique angles at which it's captured, alignment issues arise . This correction can introduce inaccuracies in the fused point cloud, causing, for instance, buildings in the point cloud to be mistakenly colored with ground features from nearby areas, or vice versa (Figure 6.5b).

Moreover, shadows from large structures in the MSI often project onto the point clouds, and temporal changes in land cover can add further discrepancies. For example, agricultural fields might appear green in the point cloud but show up as harvested in the imagery, or the opposite (Figure 6.5a). When NIR data is incorporated, the effect is similar: NIR values from ground or tree features near buildings may be projected onto building walls in point cloud, leading to inconsistent NIR representation across the fused point cloud (Figure 6.4).

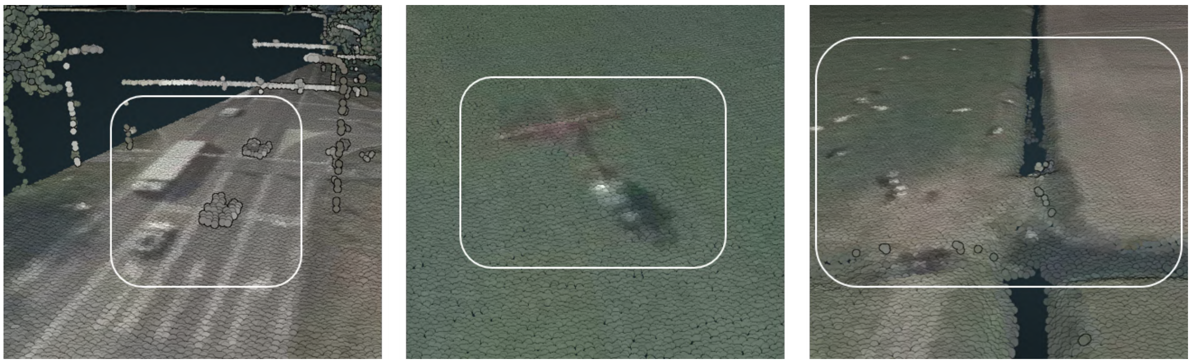
The artifacts incurred from fusing the two datasets could be a reason for the decline in performance on specific classes, and also overall. For instance, the *Buildings* class saw a decline when NIR was added as an additional feature to the deep learning models. Similarly, the *Ground* class performance dropped when RGB features were added.


(a) RGB point cloud

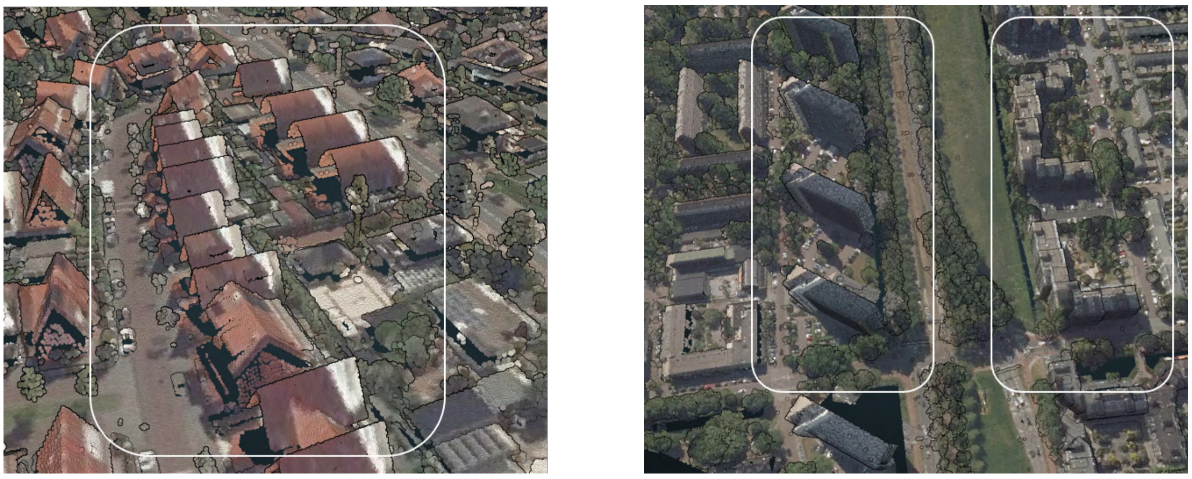


(b) Inconsistent NIR along building walls and roof

Figure 6.4: Artifacts from fusing aerial imagery NIR with AHN point cloud



(a) **Temporal Misalignment Artifacts**: Temporal differences causing vehicle and land use inconsistencies in point cloud.

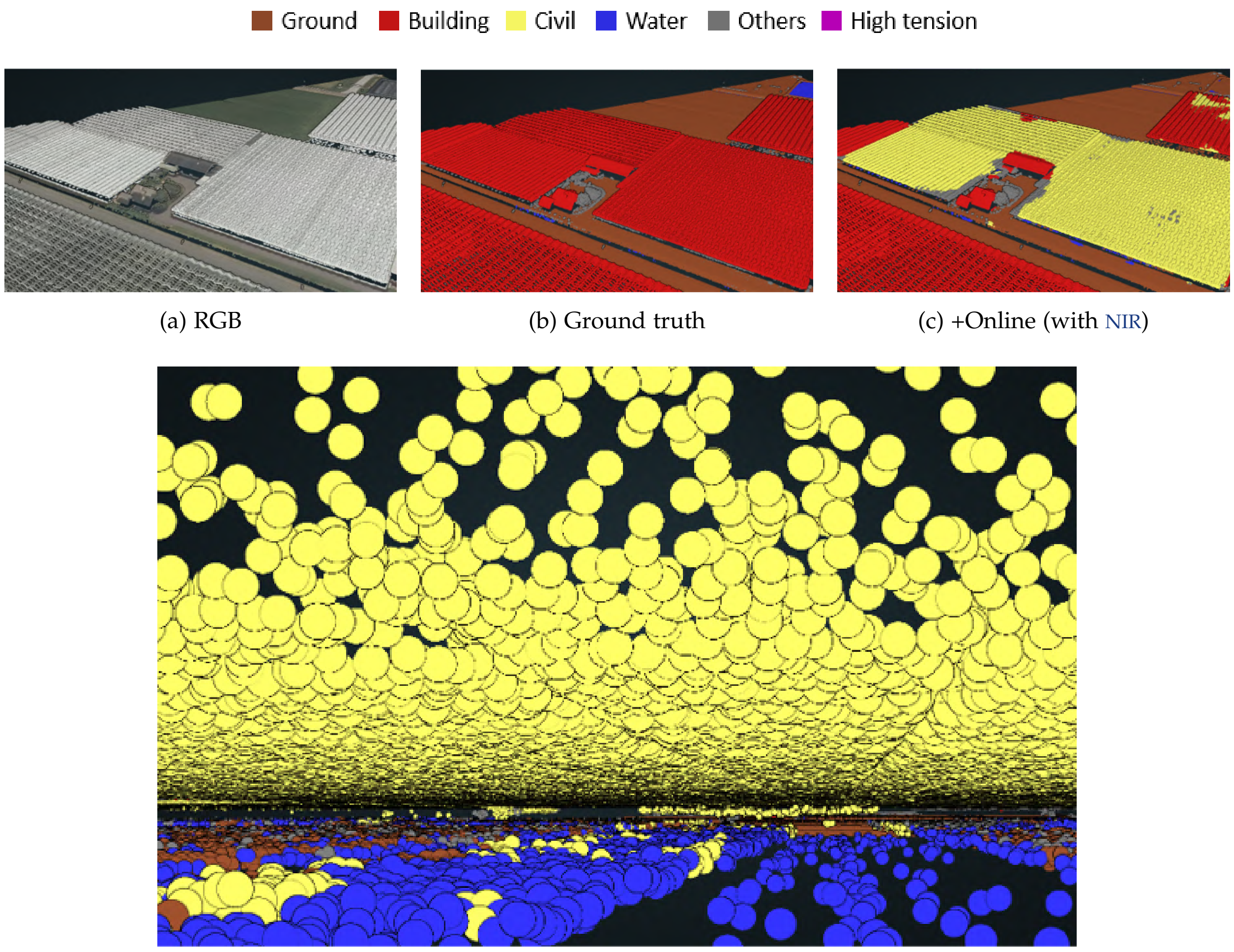


(b) **RGB Alignment Issues**: Misalignment resulting in incorrect coloring of roofs and ground surfaces

Figure 6.5: Artifacts from fusing aerial imagery RGB with AHN Point Cloud

3. Missing context in training data

A notable observation across all experiments is the stark contrast between the per-class accuracy and the IoU for *Civil structures*. While achieving more than 75% accuracy in most of experiments, with the best 84.2% when trained with additional RGB features, IoU remains less than 5%. This suggests that, while the model is correctly identifying most *Civil structures*, it is also mistakenly classifying many points from other classes of *Ground* and *Building* as *Civil*. The main reason for it being that, in the training data, there is almost no green house data for the models to learn from. Structurally, green houses look very different from normal buildings, but semantically they are labeled as *Buildings* in AHN. When the deep learning models encountered these green house structures during validation, they have to classify structures which they have not seen during the training phase. So, models created their own context to label green houses, by labeling roof as *Civil* and points under the roof as *Water* and *Ground* (Figure 6.6). This emphasizes the fact that deep learning models only learn from what they see—Good inputs gives good outputs.



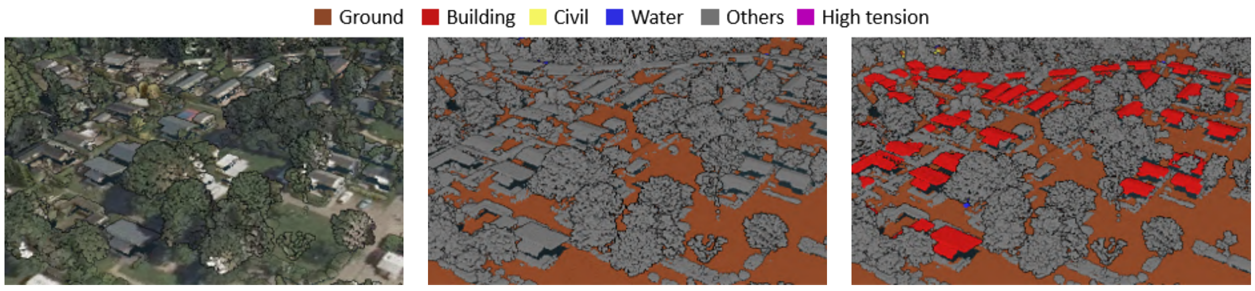
(a) RGB (b) Ground truth (c) +Online (with NIR)

(d) Under the greenhouse [+Online (with NIR)]

Figure 6.6: Greenhouses mislabeling

4. Misclassifications in ground truth

In the validation dataset, we observed numerous instances where buildings were incorrectly labeled as *Others* in the ground truth data (Figure 6.7b), indicating errors in the ground truth itself. Although the model has correctly identified these points as belonging to the *Buildings* class (Figure 6.7c), this mislabeling in the ground truth data prevents the correct predictions from being accurately reflected in metrics such as accuracy and IoU. Consequently, despite the model's strong predictive ability, these metrics suggest otherwise. Currently, aside from qualitative observations, we lack a quantitative method to measure these discrepancies.



(a) RGB (b) Ground truth (c) +Online (trained on base features)

Figure 6.7: Wrong *Building* classifications in AHN

6.2 Conclusions

In this section we will sequentially address each of the research sub-question.

- How can geospatial knowledge be incorporated into a deep learning framework, and what benefits does this integration provide?

  Geospatial knowledge is crucial to our confidence-aware deep learning model. In the preprocessing stage, each point is assigned a confidence score using *Neighborhood Consistency* based on its surroundings, which is then further refined with heuristic knowledge. The model is then trained iteratively using these confidence scores, allowing the online learning framework to continuously enhance the model's understanding, refining the training data, and improve segmentation results. Most importantly, unlike the self-training approach proposed by Li et al. [2021], which uses a computationally intensive GAN with two networks, our method is lightweight and requires significantly fewer computational resources.

- To what extent does the online learning strategy enhance the model's ability to correct misclassifications and improve overall segmentation accuracy compared to traditional training approaches?

  When only raw LiDAR data (intensity and elevation) is available for segmentation tasks, a deep learning model with the *Online* learning strategy proves to be the most effective. Specifically, the *Online* strategy using base features—intensity and elevation—achieves the highest overall accuracy. However, when additional features like NIR or RGB are available, the *Online* model falls short of the *Baseline* model, both in overall performance and in specific classes such as *High tension* and *Civil*. Observing the *Online* label updates on the training data, the updated labels show qualitative improvements, with noisy labels and inconsistencies rectified, and bulk misclassifications, such as *Buildings* mislabeled as *Others*, are corrected. This suggests that the *Online* strategy can enhance the quality of training data; however, these improvements cannot be easily quantified.

- What is the impact of incorporating additional spectral features (such as NIR and RGB) on the performance of the proposed confidence-aware deep learning model for point cloud segmentation?

  NIR offers marginal improvements for specific classes like *Others* (e.g., vegetation) due to its sensitivity to plant characteristics, though it introduces spectral inconsistencies that can reduce accuracy for classes like *Building*. While both NIR and RGB significantly improve the *Baseline* model's performance, they fail to yield similar gains when combined with the *Online* learning strategy. Artifacts and temporal misalignments, likely explain the performance dips seen in *Building* (when using NIR) and *Ground* (when using RGB) classifications, as these inconsistencies affect spatial and spectral alignment. Based on the specific requirements and characteristics of each case, the appropriate choice of additional features (NIR or RGB) can be identified to optimize model performance.

6.3 Future scope

1. A couple of limitations discussed in this thesis (Chapter 6.1.4) could be addressed by employing transformer networks, with the availability of greater computational resources to train the models. Additionally, the colored point cloud data provided by Geotiles presents quality issues, such as temporal and alignment artifacts. An alternative approach would be to use the raw AHN point cloud data and independently fuse it with aerial or satellite imagery acquired as close as possible to the point cloud data collection date. This method could mitigate alignment issues and improve the coherence and reliability of the supplementary features, such as NIR and RGB.

2. In theory, our approach is designed to be generic and applicable to various point cloud data sources. However, the effectiveness of the confidence scoring mechanism and online learning strategy has not yet been verified on other types of point cloud data, such as Terrestrial Laser Scanning (TLS) or Mobile Laser Scanning (MLS), or across different geographic regions. Experimenting with TLS and MLS data would further validate the generalizability of our method to diverse data types and environments.

3. Another promising direction is to leverage recent advances in computer vision, such as the SAM2 model [Ravi et al., 2024], for extracting building footprints directly from images, which could also be utilized to refine confidence scores and improve overall accuracy. A potential future direction involves incorporating synthetic point cloud data [Shinohara et al., 2021] to address the performance limitations of minority classes. By artificially generating additional data for underrepresented categories, such as *high tension* lines and *civil* structures, the model could be exposed to a more balanced dataset, improving its ability to learn robust features. This approach would not only mitigate the effects of class imbalance but could also lead to more accurate and reliable classification of these challenging categories, enhancing overall model performance and generalizability.

Bibliography

AHN (2024a). Kwaliteitsbeschrijving. https://www.ahn.nl/kwaliteitsbeschrijving. Accessed: 2024-06-05.

AHN (2024b). Producten. https://www.ahn.nl/producten. Accessed: 2024-06-05.

Bello, S. A., Yu, S., and Wang, C. (2020). Review: deep learning on 3d point clouds.

Bergmann, D. (2023). What is semi-supervised learning? https://www.ibm.com/topics/semi-supervised-learning. Accessed: 2024-04-25.

Cai, Y. (2024). Splitsfc: A database solution for massive point cloud data management. Master's thesis, Delft University of Technology, Delft, The Netherlands.

Delft High Performance Computing Centre (DHPC) (2024). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2.

Donkers, T. (2024). Assessing the quality of lidar infrastructure point clouds. Master's thesis, Delft University of Technology, Delft, The Netherlands.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.

Elshehaby, A. R. and Taha, L. G. E.-d. (2009). A new expert system module for building detection in urban areas using spectral information and lidar data. *Applied Geomatics*, 1(4):97–110.

ESRI (2023). What is intensity data? https://desktop.arcgis.com/en/arcmap/latest/manage-data/las-dataset/what-is-intensity-data-.htm. Accessed: 2024-09-12.

Fernandez-Diaz, J., Carter, W., Shrestha, R., and Glennie, C. (2014). Now you see it… now you don't: Understanding airborne mapping lidar collection and data product generation for archaeological research in mesoamerica. *Remote Sensing*, 6(10):9951–10001.

Griffiths, D. and Boehm, J. (2019). Weighted point cloud augmentation for neural network training data class-imbalance. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W13:981–987.

Grilli, E., Farella, E. M., Torresani, A., and Remondino, F. (2019). Geometric features analysis for the classification of cultural heritage point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W15:541–548.

He, Y., Yu, H., Liu, X., Yang, Z., Sun, W., and Mian, A. (2021). Deep learning based 3d segmentation: A survey.

Hu, Q., Yang, B., Fang, G., Guo, Y., Leonardis, A., Trigoni, N., and Markham, A. (2021). Sqn: Weakly-supervised semantic segmentation of large-scale 3d point clouds.

Hua, B.-S., Tran, M.-K., and Yeung, S.-K. (2017). Pointwise convolutional neural networks.

Huang, X. and Zhang, L. (2012). Morphological building/shadow index for building extraction from high-resolution imagery over urban areas. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(1):161–172.

Jie Shan, C. K. T., editor (2018). *Topographic Laser Ranging and Scanning: Principles and Processing*. CRC Press.

Jing, L. and Tian, Y. (2019). Self-supervised visual feature learning with deep neural networks: A survey.

Kanezaki, A., Matsushita, Y., and Nishida, Y. (2016). Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints.

Kappé, J. (2024). Deep learning-based segmentation of cracks within a photogrammetry solution. Master's thesis, Delft University of Technology, Delft, The Netherlands.

Lai, X., Liu, J., Jiang, L., Wang, L., Zhao, H., Liu, S., Qi, X., and Jia, J. (2022). Stratified transformer for 3d point cloud segmentation.

Landrieu, L., Raguet, H., Vallet, B., Mallet, C., and Weinmann, M. (2017). A structured regularization framework for spatially smoothing semantic labelings of 3d point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 132:102–118.

Li, H., Sun, Z., Wu, Y., and Song, Y. (2021). Semi-supervised point cloud segmentation using self-training with label confidence prediction. *Neurocomputing*, 437:227–237.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017). Focal loss for dense object detection.

Madanu, S. C. (2024). Automatic misclassifications identification of semantically segmented ahn5 point cloud. `https://github.com/AutumnMoon00/RefineNet/blob/main/AHN-errors-internship-report/Internship_Report_AHN_Errors.pdf`. Accessed: 2024-10-22.

Maturana, D. and Scherer, S. (2015). Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE.

Mazzari, V. (2019). What is lidar technology? `https://www.generationrobots.com/blog/en/what-is-lidar-technology/`. Accessed: 2024-07-30.

Misra, I., Girdhar, R., and Joulin, A. (2021). An end-to-end transformer model for 3d object detection.

Niemeyer, J., Rottensteiner, F., and Soergel, U. (2012). Conditional random fields for lidar point cloud classification in complex urban areas. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 1:263–268.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, , Shazeer, N., Ku, A., and Tran, D. (2018). Image transformer.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library.

Petrie, G. (2011). Airborne topographic laser scanners. *GEO: connexion International Magazine*, 10(1):28–31.

Pettorelli, N. (2013). *The Normalized Difference Vegetation Index*. Oxford University Press.

Poux, F. and Billen, R. (2019). Voxel-based 3d point cloud semantic segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*, 8(5):213.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2016). Pointnet: Deep learning on point sets for 3d classification and segmentation.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space.

Rabbani Shah, T., van den Heuvel, F., and Vosselman, M. (2006). Segmentation of point clouds using smoothness constraint. In Maas, H.-G. and Schneider, D., editors, *Proceedings of the ISPRS Com. V Symposium*, pages 248–253. Dresden University of Technology. ISPRS Symposium: Image Engineering and Vision Metrology ; Conference date: 25-09-2006 Through 27-09-2006.

Ravi, N., Gabeur, V., Hu, Y.-T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., Mintun, E., Pan, J., Alwala, K. V., Carion, N., Wu, C.-Y., Girshick, R., Dollár, P., and Feichtenhofer, C. (2024). Sam 2: Segment anything in images and videos.

Riegler, G., Ulusoy, A. O., and Geiger, A. (2016). Octnet: Learning deep 3d representations at high resolutions.

Robert, D., Raguet, H., and Landrieu, L. (2023). Efficient 3d semantic segmentation with superpoint transformer.

Schowengerdt, R. A. (2007). *Remote Sensing: Models and Methods for Image Processing*. Elsevier; Academic Press, Amsterdam, Netherlands; Burlington, MA, 3rd edition.

Schutz, M., Ohrhallinger, S., and Wimmer, M. (2020). Fast out-of-core octree generation for massive point clouds. *Computer Graphics Forum*, 39(7):13.

Shinohara, T., Xiu, H., and Matsuoka, M. (2021). 3d point cloud generation using adversarial training for large-scale outdoor scene. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, volume 29, pages 2935–2938. IEEE.

Siegmund, A. and Menz, G. (2005). Fernes nah gebracht - satelliten- und luftbildeinsatz zur analyse von umweltveränderungen im geographieunterricht. *Geographie und Schule*, 154:2–10.

Strudel, R., Garcia, R., Laptev, I., and Schmid, C. (2021). Segmenter: Transformer for semantic segmentation.

Su, H., Maji, S., Kalogerakis, E., and Learned-Miller, E. (2015). Multi-view convolutional neural networks for 3d shape recognition.

Sun, S., Yue, X., Bai, S., and Torr, P. (2021). Visual parser: Representing part-whole hierarchies with transformers.

Thomas, H., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Gall, Y. L. (2018). Semantic classification of 3d point clouds with multiscale spherical neighborhoods.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds.

van der Heide, D., van Natijne, A., Alkemade, I., and Hulskemper, D. (2024). WP1: Inventarisatie van puntenwolken in Nederland. techreport, Rijkswaterstaat, Het Waterschapshuis, Kadaster TU Delft.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need.

Weinmann, M., Jutzi, B., Hinz, S., and Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286–304.

Wu, K., Peng, H., Chen, M., Fu, J., and Chao, H. (2021). Rethinking and improving relative position encoding for vision transformer.

Xie, Y., Sha, Z., and Yu, M. (2008). Remote sensing imagery in vegetation mapping: a review. *Journal of Plant Ecology*, 1(1):9–23.

Xie, Y., Tian, J., and Zhu, X. X. (2020). Linking points with labels in 3d: A review of point cloud semantic segmentation. *IEEE Geoscience and Remote Sensing Magazine*, 8(4):38–59.

Xu, S., Wan, R., Ye, M., Zou, X., and Cao, T. (2022). Sparse cross-scale attention network for efficient lidar panoptic segmentation.

Xu, Y., Fan, T., Xu, M., Zeng, L., and Qiao, Y. (2018). Spidercnn: Deep learning on point sets with parameterized convolutional filters.

Yao, S., Guan, R., Huang, X., Li, Z., Sha, X., Yue, Y., Lim, E. G., Seo, H., Man, K. L., Zhu, X., and Yue, Y. (2024). Radar-camera fusion for object detection and semantic segmentation in autonomous driving: A comprehensive review. *IEEE Transactions on Intelligent Vehicles*, 9(1):2094–2128.

Ye, S., Chen, D., Han, S., and Liao, J. (2022). Robust point cloud segmentation with noisy annotations.

Zhan, Q. and Yu, L. (2012). *Segmentation of LiDAR Point Cloud Based on Similarity Measures in Multi-dimension Euclidean Space*, pages 349–357. Springer Berlin Heidelberg.

Zhang, C., Wan, H., Shen, X., and Wu, Z. (2021). Pvt: Point-voxel transformer for point cloud learning.

Zhang, J., Lin, X., and Ning, X. (2013). Svm-based classification of segmented airborne lidar point clouds in urban areas. *Remote Sensing*, 5(8):3749–3775.

Zhang, R., Wu, Y., Jin, W., and Meng, X. (2023). Deep-learning-based point cloud semantic segmentation: A survey. *Electronics*, 12(17):3642.

Zhao, H., Jia, J., and Koltun, V. (2020a). Exploring self-attention for image recognition.

Zhao, H., Jiang, L., Jia, J., Torr, P., and Koltun, V. (2020b). Point transformer.

Zheng, M., Gao, P., Zhang, R., Li, K., Wang, X., Li, H., and Dong, H. (2020). End-to-end object detection with adaptive clustering transformer.

Zhou, J., Xiong, Y., Chiu, C., Liu, F., and Gong, X. (2023). Sat: Size-aware transformer for 3d point cloud semantic segmentation.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.