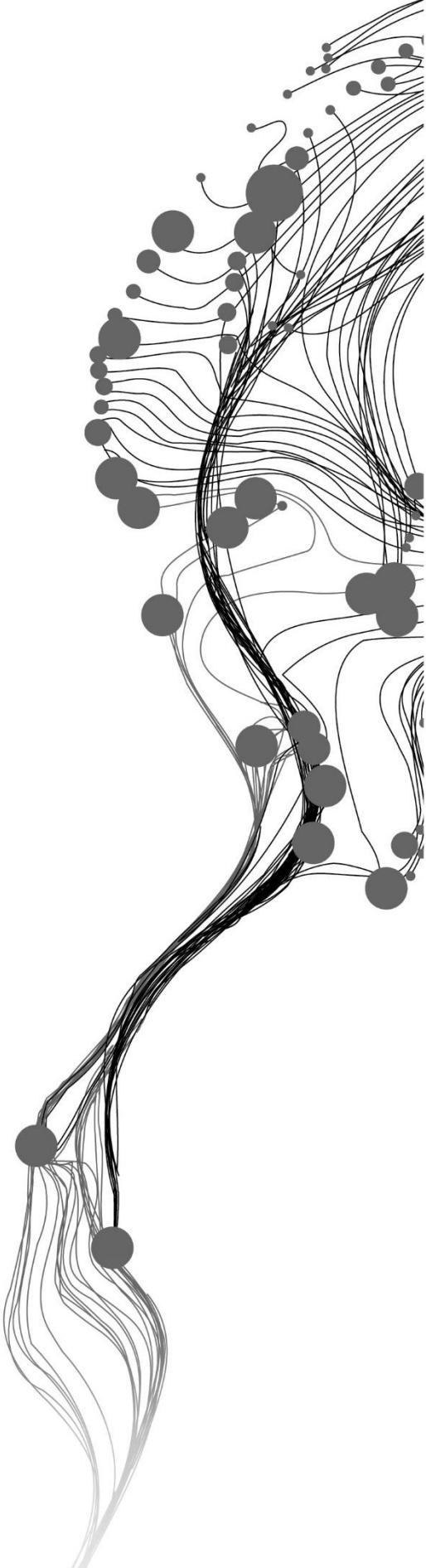


Deep Learning-based Change Detection and Classification for Airborne Laser Scanning Data

JORGES NOFULLA
June, 2023

SUPERVISORS:
DR.IR. S.J. OUDE ELBERINK (SANDER)
DR. HABIL. Y. YANG (MICHAEL)



Deep Learning-based Change Detection and Classification for Airborne Laser Scanning Data

JORGES NOFULLA
Enschede, The Netherlands

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: Geoinformatics

SUPERVISORS:

DR.IR. S.J. OUDE ELBERINK (SANDER)
DR. HABIL. Y. YANG (MICHAEL)

THESIS ASSESSMENT BOARD CHAIR:
PROF. DR. IR. GEORGE VOSSELMAN

EXTERNAL ASSESSOR:
DR. R.C. LINDENBERGH (TU DELFT)

DISCLAIMER

This document describes work undertaken as part of a programme of study at the Faculty of Geo-Information Science and Earth Observation of the University of Twente. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the Faculty.

ABSTRACT

Urban change detection plays a critical role in many domains such as city planning, infrastructure development, risk assessment, and land-use planning. However, the accurate classification of different types of changes in a 3D urban environment remains a challenging task. Conventional methods, which typically involve transforming the data into a DSM or Voxels, often fall short in managing the complexity of point clouds. On the other hand, more complex deep learning models have shown promise but still face challenges in real-world applications.

In this Master's thesis, we address these challenges by implementing and evaluating three models: Random Forest, Fully Connected Neural Network, and Convolution Neural Network. Our approach builds on the foundational success of 2D change detection methods. We are confident that these proven techniques when adapted and extended to the 3D realm, can provide a simplified, efficient, and precise detection method. We utilize two datasets: the simulated Urb3DCD dataset, which provides a variety of class labels representing different types of changes, and the real-world AHN dataset from the Netherlands, offering a real and complex urban landscape. Our research provides a method that could be beneficial for urban planning, infrastructure development, and hazard and risk assessment.

Our research begins with an explanation of the pre-processing of the data, chosen models, and their implementation. We train and validate these models on our datasets, assessing their ability to accurately classify different types of changes. Our findings reveal that while all models demonstrate strong performance, each has its strengths and weaknesses. The RF model, for instance, excels in areas that are well-represented in the training data, while the deep learning models display superiority in differentiating between similar classes. The CNN model adds an extra layer of understanding the data by accounting for spatial relationships between points, thus enhancing overall accuracy. The results obtained from the simulated dataset were impressive. We then apply these models to the AHN dataset, revealing that the model's performance is highly sensitive to feature selection, the quality of training data, and how representative the data is. The direct comparison of classes between the simulated and real datasets indicates superior results from the simulated dataset, suggesting the need for better-quality training data for real-world applications.

One significant insight from our work is the demonstrated potential to leverage simpler methods for processing point cloud data, thus bypassing the need for more complex and computationally intensive techniques. Furthermore, we shed light on the challenges encountered when applying these models to real-world scenarios. This thesis, therefore, not only contributes to the existing body of knowledge on urban change detection using 3D point cloud data but also opens up new avenues for future research.

Keywords: Deep Learning, Point Clouds, Change Detection, Aerial Laser Scanning (ALS), Urban Environment, Machine Learning, Actueel Hoogtebestand Nederland (AHN), Urb3DCD dataset

ACKNOWLEDGEMENTS

I want to start by expressing my sincere gratitude to my main supervisor, Dr. Ir. S.J. Oude Elberink, for his guidance, patience, knowledge and, continuous support throughout this process. This helped me improve both academically and personally. I could not have asked for a greater mentor and I truly appreciate your effort and dedication to this research.

I also want to extend my appreciation to my second supervisor, Dr. Habil. Y. Yang, whose constructive feedback and suggestions helped me improve my work.

I would also like to thank Nishit Patel, a good friend of mine. You gave me inspiration and kept me motivated with your emotional support, our common interests, and the many brainstorming sessions we had.

A special note of gratitude goes to the Municipality of Amsterdam, where I had the opportunity to do an internship. I gained invaluable knowledge from this practical experience. I am particularly thankful to Daan Bloembergen, Nico de Graaff, and Iva Gronishka. I would also like to acknowledge my fellow interns at the Municipality, whose friendship and experiences enriched my internship.

Finally, this journey would not have been possible without the support of my family and friends back home. Your faith in my abilities, constant support, and unconditional love have given me courage and strength. I can never thank you enough.

In conclusion, while this thesis carries my name, it is a product of the combined efforts of all the people that supported me through this journey. My sincere gratitude goes out to all those who have supported me in reaching this significant academic milestone.

Thank you.

TABLE OF CONTENTS

| | | |
|--------|---|----|
| 1. | Introduction..... | 7 |
| 1.1. | Motivation and applications | 7 |
| 1.2. | Research objectives and questions | 9 |
| 1.2.1. | Main objective | 9 |
| 1.2.2. | Sub-objective (1):..... | 9 |
| 1.2.3. | Sub-objective (2):..... | 9 |
| 1.2.4. | Sub-objective (3) | 9 |
| 1.3. | Contributions | 10 |
| 2. | Related works and State of the art | 10 |
| 2.1. | 2D change detection | 11 |
| 2.2. | 3D change detection | 15 |
| 2.3. | Research Gap | 20 |
| 3. | Datasets | 21 |
| 3.1. | AHN dataset..... | 21 |
| 3.2. | URB3DCD dataset..... | 23 |
| 4. | Methodology and algorithms | 26 |
| 4.1. | Concept and Approach | 27 |
| 4.2. | Generation of Training Data..... | 28 |
| 4.3. | Change detection models..... | 32 |
| 4.3.1. | Change Detection using Random Forest Classifier..... | 32 |
| 4.3.2. | Change Detection using Fully Connected Neural Network | 33 |
| 4.3.3. | Change Detection using Convolutional Neural Network | 36 |
| 4.4. | Experimental Design and Parameters..... | 38 |
| 5. | Results and Analysis | 38 |
| 5.1. | Results for URB3DCD Dataset..... | 39 |
| 5.1.1. | Random Forest Classifier..... | 39 |
| 5.1.2. | Fully Connected Neural Network | 42 |
| 5.1.3. | Convolutional Neural Network | 45 |
| 5.2. | Results for AHN Dataset..... | 48 |
| 5.2.1. | Random Forest Classifier..... | 48 |
| 5.2.2. | Fully Connected Neural Network | 50 |
| 5.2.3. | Convolutional Neural Network | 52 |
| 6. | Discussion..... | 53 |
| 6.1. | Comparative Analysis and Visualization of our Results | 53 |
| 6.1.1. | URB3DCD Dataset..... | 53 |
| 6.1.2. | AHN dataset..... | 58 |
| 6.1.3. | Comparison between URB3DCD and AHN Datasets | 63 |
| 6.2. | Comparison with the state of the art (Urb3DCD) | 66 |
| 6.3. | Comparison with the state of the art (AHN) | 68 |
| 7. | Conclusions and Final remarks..... | 69 |
| 8. | Appendix..... | 70 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1 - AHN acquisition methods..... | 8 |
| Figure 2 - Definition of different change types..... | 10 |
| Figure 3 - (a) Aerial photograph of heterogeneous landscape (b) fine-scale segmentation (c) coarse-scale segmentation (d) object-based classification of woody cover, resulting in 97% accuracy..... | 12 |
| Figure 4 – Flow diagram of the change detection model | 13 |
| Figure 5 - Change detection using ANN..... | 14 |
| Figure 6 - MSCANet architecture overview | 15 |
| Figure 7 - Change detection results, using Octrees | 16 |
| Figure 8 - Overview of M3C2 algorithm..... | 16 |
| Figure 9 - Comparing the geometry of two point clouds using voxels | 17 |
| Figure 10 - Workflow of DSM change detection method..... | 17 |
| Figure 11 - The complete overview of the algorithm workflow (Cserép and Lindenbergh) | 18 |
| Figure 12 - PointNet Architecture | 19 |
| Figure 13 - Siamese KPConv network architecture..... | 19 |
| Figure 14 - Data acquisition years for AHN3 data (left) and planned years of data acquisition for AHN4 (right) | 21 |
| Figure 15 - AHN dataset locations..... | 22 |
| Figure 16 - Noise with the labeling of Urb3DCD data. It is noticeable that points that do not represent a change are labeled as removed buildings (yellow points). | 23 |
| Figure 17 - One of the training tiles of the Urb3DCD dataset..... | 24 |
| Figure 18 - The location of the simulated datasets (Lyon, France)..... | 24 |
| Figure 19 - One of the validation tiles of the Urb3DCD dataset..... | 25 |
| Figure 20 - One of the testing tiles of the Urb3DCD dataset | 25 |
| Figure 21 - An overview of the methodology | 26 |
| Figure 22 - An (a) AHN3 tile and (b) AHN4 tile used for training | 27 |
| Figure 23 - A balanced AHN tile used for training. Red is for new buildings, yellow is for removed buildings and blue is unchanged. | 29 |
| Figure 24 - Target Label Generation..... | 29 |
| Figure 25 - A closer look into the data cleaning. Original (up) and cleaned (down)..... | 30 |
| Figure 26 - AHN training tile before (left) and after (right) the point cleaning..... | 30 |
| Figure 27 - AHN tiles, where (a) and (b) are the same area in the AHN3 with RGB from different points of view. (c) and (d) are also the same area but in AHN4..... | 31 |
| Figure 28 - The model of a Random Forest algorithm used for classification | 33 |
| Figure 29 - Our FCNN model | 35 |
| Figure 30 - A summary of the process used for the CNN models | 36 |
| Figure 31 - Our CNN model..... | 37 |
| Figure 32 - Confusion matrix of RF model trained on the full dataset..... | 40 |
| Figure 33 - Per-class accuracy of all datasets for the RF model, Urb3DCD | 41 |
| Figure 34 - Confusion matrix of FCNN model trained on the full dataset..... | 43 |
| Figure 35 - Per-class accuracy of all datasets for the FCNN model, Urb3DCD | 44 |
| Figure 36 - Confusion matrix of CNN model trained on the full dataset | 46 |
| Figure 37 - Per-class accuracy of all datasets for the CNN model, Urb3DCD | 47 |
| Figure 38 - Per-class accuracy for the RF model, AHN dataset..... | 49 |
| Figure 39 - Confusion matrix of RF model trained on the AHN dataset..... | 49 |

| | |
|--|----|
| Figure 40 - Confusion matrix of FCNN model trained on the AHN dataset..... | 51 |
| Figure 41 - Per-class accuracy for the FCNN model, AHN dataset..... | 51 |
| Figure 42 - Confusion matrix of CNN model trained on the AHN dataset | 52 |
| Figure 43 - Per-class accuracy for the CNN model, AHN dataset..... | 53 |
| Figure 44 - Per-class accuracy of all the models, Urb3DCD..... | 54 |
| Figure 45 - A predicted tile from the Urb3DCD dataset. (a) Ground truth, (b) Random forest, (c) Fully Connected NN, and (d) Convolution NN | 56 |
| Figure 46 - Close up view of Urd3DCD dataset results. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN | 57 |
| Figure 47 – A close look into a predicted test tile of the AHN dataset. (a) Original CNN output and (b) Cleaned CNN output..... | 58 |
| Figure 48 - Per-class accuracy of all the models, AHN dataset | 59 |
| Figure 49 - Prediction on an AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN | 60 |
| Figure 50 - A closer look into the prediction of the AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN | 60 |
| Figure 51 - Prediction on another AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN | 61 |
| Figure 52 - A closer look at the prediction of the tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN..... | 61 |
| Figure 53 - Predicted AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN | 62 |
| Figure 54 - Comparison of AHN results with Urb3DCD. RF model..... | 64 |
| Figure 55 - A closer look at the densities of our datasets. Where (a) is an AHN and (b) a Urb3DCD tile | 64 |
| Figure 56 - Comparison of AHN results with Urb3DCD. FCNN model..... | 65 |
| Figure 57 - Comparison of AHN results with Urb3DCD. CNN model | 65 |
| Figure 58 - Confusion matrix of RF model trained on the first half dataset..... | 70 |
| Figure 59 - Confusion matrix of RF model trained on the second half dataset..... | 71 |
| Figure 60 - Confusion matrix of FCNN model trained on the first half dataset | 72 |
| Figure 61 - Confusion matrix of FCNN model trained on the second half dataset..... | 73 |
| Figure 62 - Confusion matrix of CNN model trained on the first half dataset..... | 74 |
| Figure 63 - Confusion matrix of CNN model trained on the second half dataset | 75 |

LIST OF TABLES

| | |
|--|----|
| Table 1 - AHN dataset properties | 22 |
| Table 2 - AHN dataset accuracies | 22 |
| Table 3 - Urb3DCD dataset properties | 23 |
| Table 4 - Urb3DCD classes distribution..... | 39 |
| Table 5 - Accuracies of RF model trained with the full dataset | 39 |
| Table 6 - Accuracies of FCNN model trained with the full dataset | 42 |
| Table 7 - Accuracies of CNN model trained with the full dataset..... | 45 |
| Table 8 - AHN classes distribution..... | 48 |
| Table 9 -Accuracies of RF model trained on the AHN dataset | 48 |
| Table 10 - Accuracies of FCNN model trained on the AHN dataset..... | 50 |
| Table 11 - Accuracies of CNN model for AHN dataset..... | 52 |
| Table 12 - Validation datasets classes distribution..... | 63 |
| Table 13 -Per-class IoU scores on Urb3DCD–V2 low density LiDAR dataset..... | 66 |
| Table 14 - Per-class IoU scores on the AHN LiDAR dataset..... | 68 |
| Table 15 - Accuracies of RF model trained with the first half of the dataset..... | 70 |
| Table 16 - Accuracies of RF model trained with the second half of the dataset | 71 |
| Table 17 - Accuracies of FCNN model trained with the first half of the dataset | 72 |
| Table 18 - Accuracies of RF model trained with the second half of the dataset | 73 |
| Table 19 - Accuracies of CNN model trained with the first half of the dataset..... | 74 |
| Table 20- Accuracies of CNN model trained with the second half of the dataset..... | 75 |

1. INTRODUCTION

1.1. Motivation and applications

The process of detecting changes in an object's or phenomenon's condition through repeated observation is referred to as change detection (Singh, 1989). Due to limitations in technology and resolution, change detection has historically been used to monitor changes in agricultural and land cover (Guerin, Binet, & Pierrot-Deseilligny, 2014). With the rapid development of data acquisition technologies in recent years, the scope of change detection has broadened, enabling its use in diverse fields such as water conservancy, disaster evaluation, illegal building detection, urban change documentation, construction site surveying, and city map/model updating (Hebel et al., 2013).

The growing need for efficient city administration and frequent updates to 3D city models in response to rapid urban expansion calls for innovative and effective change detection methods (United Nations, 2008). Urban change detection plays a vital role in addressing the challenges associated with urban growth, promoting sustainable development, and maintaining a high quality of life. By swiftly identifying changes in urban areas, spatial planning, damage estimation in natural disasters, and up-to-date city models can be ensured (Tran et al., 2018).

Traditional change detection methods use remotely sensed multi-spectral or optical 2D images, LIDAR (light detection and ranging), or RADAR data. While 2D images offer simplicity and reduced computational demands, they lack height information and are sensitive to environmental conditions, leading to measurement uncertainties (Yadav et al., 2022). In contrast, 3D point clouds (PCs) generated by LIDAR systems provide essential height information and preserve the original geometric information in 3D space, making them more suitable for urban areas (Guo et al., 2020).

This research takes place in the Netherlands and uses it's available point cloud as one of the datasets ([GeoTiles.nl](#)). The second dataset that we use in this research is a simulated point cloud generated by de Gelis et al., 2023. The most common methods for obtaining point clouds in the Netherlands include photogrammetry (dense image matching), MLS (mobile laser scanning), and ALS (aerial laser scanning).

The photogrammetry techniques use multiple images to create a 3D point cloud of the area by dense matching. While photogrammetry techniques are cost-effective and easily achievable, they exhibit lower accuracy and are prone to occlusion, low resolution, noise, and they are vulnerable to a lot of problems when it comes to studying a 3D environment (Remondino et al., 2014) (W. Liu et al., 2019)(Lehtola et al., 2017). Laser scanners (active sensors) offer an alternative for generating 3D point clouds with high capture speed and point density (Vosselman & Maas, 2010).

The Netherlands' up-to-date height model, Actueel Hoogtebestand Nederland (AHN), is a digital terrain model produced jointly by 26 water boards and Rijkswaterstaat. AHN primarily uses aerial laser scanning, with measurements taken from helicopters and aircraft (Figure 1). Currently, in its fourth generation (2020-2022), the AHN has seen increased accuracy and point cloud density with each iteration. However, data quality may vary across the country due to different acquisition processes. Despite the availability of rapid and reliable 3D point cloud acquisition methods, keeping city models updated for sustainable urban planning remains challenging due to the accelerated pace of urban change. Traditional and manual methods are insufficient for maintaining up-to-date government databases.

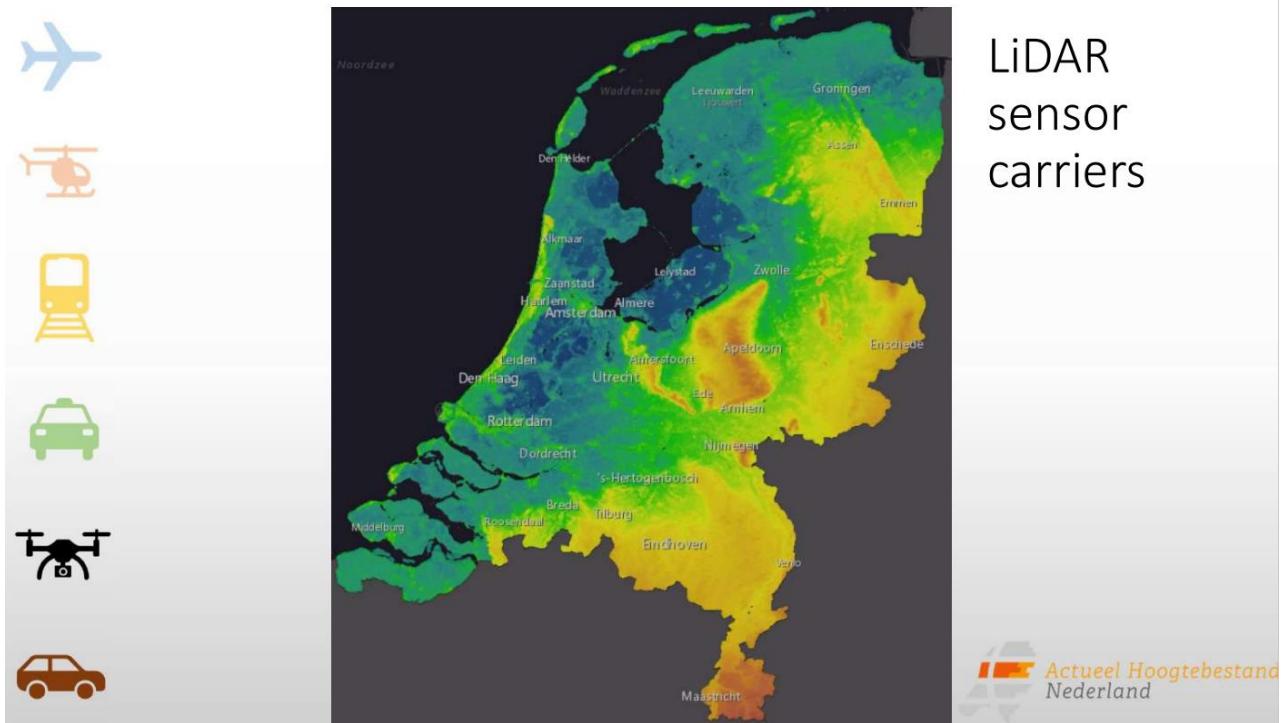


Figure 1 - AHN acquisition methods

[Image reference](#)

In this thesis, we will explore and develop two types of deep learning methods (FCNN and CNN) and one machine learning method (Random Forest), to address the challenges of urban change detection. These models aim to provide a more accurate and efficient solutions for updating city models, detecting changes, and classifying them in urban areas.

To evaluate the performance of our proposed models, we will utilize two distinct datasets: the Actueel Hoogtebestand Nederland (AHN) data and the [URB3DCD](#) dataset. The AHN data, provides point cloud information with labelled classes such as ground, buildings etc. But because of the lack of classes and data, we could extract only 3 type of change classes for our algorithm. In contrast, the URB3DCD dataset is a more comprehensive resource containing point clouds with seven different change class labels. By incorporating both datasets, we aim to thoroughly test the effectiveness of the deep learning methods across a range of urban settings and classification tasks.

Our research will contribute to the development of automated urban change detection solutions by leveraging the capabilities of deep learning methods in a raw point cloud dataset. By incorporating state-of-the-art deep learning techniques and diverse datasets, this thesis aims to advance the field of urban change detection and support sustainable urban planning efforts worldwide. This research is inspired by the principles of the 2D change detection methods, with the aim of using the accumulated expertise in this field to build a model that accurately classifies and detect changes occurring in three dimensions.

1.2. Research objectives and questions

1.2.1. Main objective

The main objective of this research is to develop a deep learning change detection and classification method for urban areas that uses as its only input point clouds. The research aims to craft an algorithm that makes the best use of the point clouds data format by detecting changes in urban areas and classifying them through a single network pipeline. The model detects changes and classifies them for the targeted classes, and is easily adaptable for every urban area of our choice.

1.2.2. Sub-objective (1):

Design and implement a state-of-the-art DL change detection algorithm that works with raw point cloud data.

Literature Review based questions:

- What is the size and type of training data required when using AHN data for change detection?
- How to prepare and build the dataset for training, testing, and validation for the algorithm?
- What backbone/main algorithm should be used when detecting changes in 3D raw PC?
- What hidden layers and activation functions should we use for our algorithm?
- What should be the features and target labels to input on our CNN?
- Can we use the idea behind 2D change detection and 3D segmentation to develop the algorithm of our research?

Research-based questions:

- Is the CNN architecture suitable for our purpose and dataset?
- How to build the model in order to process raw point cloud data?

1.2.3. Sub-objective (2):

Analyse the model's performance under various scenarios.

Research-based questions:

- What is the accuracy of detecting changes for each urban element? What is the best and worst detected object and why?
- How well does the algorithm performs with different point cloud density/ different areas of the Netherlands?
- How does the model performs in a real dataset compared to a simulated dataset? What are the problems that occur?

1.2.4. Sub-objective (3)

Evaluate the model compatibility with state-of-the-art.

Research-based questions:

- Is this method efficient when it comes to computational time and system requirements?
- How does the algorithm rank in comparison with other methods?

1.3. Contributions

This research explores the potential of point cloud data and its relationship with deep learning methods to advance urban change detection. We aim to elevate three-dimensional change detection by utilizing the pre-existing knowledge and proficiency gained from the two-dimensional change detection. Our contributions are as follows:

I) We propose a novel approach to 3D change detection that prevents data loss during pre-processing, ensuring that valuable information is preserved throughout the process.

II) We develop a versatile change detection algorithm capable of processing raw point cloud data, making it applicable across diverse urban environments.

III) Our method avoids complex data transformations and heavy calculations in the pre-processing, allowing us to maximize the utility of point cloud data by employing fast straightforward pre-processing techniques such as KDtrees.

IV) The algorithm we have designed is fast, reliable, and adaptable to various noise levels and class expansions, offering an accessible, editable, and user-friendly solution for urban change detection tasks.

V) Our innovative technique propels the field of raw point cloud-based 3D change detection forward. To foster further research and collaboration, we will make all source codes publicly available: <https://github.com/JorgesNofulla/Point-Cloud-Uraban-Change-detection>

To validate the effectiveness of our deep learning-based approach, we will utilize the multi-temporal AHN point cloud dataset (AHN3-AHN4) and the URB3DCD dataset for detecting urban changes in various urban areas. By combining cutting-edge deep learning techniques with simplistic pre-processing techniques on point cloud data, our research aims to significantly contribute to the field of urban change detection, supporting sustainable urban planning and development.

2. RELATED WORKS AND STATE OF THE ART

Change detection methods can be generally categorized as 2D, 3D, or hybrid approaches. These methods process input data from two distinct epochs, utilizing either 2D images, 3D point clouds, or a combination of both formats. In this section, we provide an overview of these different methods, discussing their underlying principles and applications.

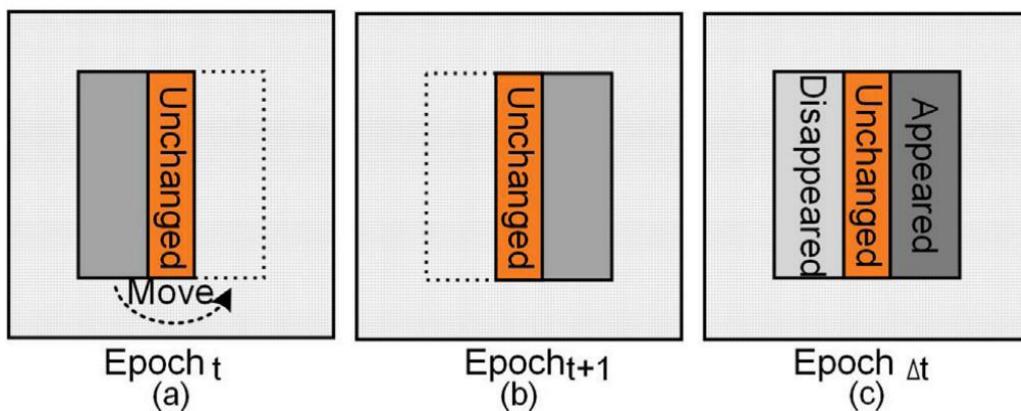


Figure 2 - Definition of different change types

[Image reference](#)

2.1. 2D change detection

The fundamental idea behind implementing 2D images for change detection is that changes in reflectance, value, or texture will result in a detected change in the objects of interest. These methods don't take into consideration height information. Some of the 2D change detection methods are :

1. **2D image differencing change detection** is one of the early change detection methods. It is easy to interpret and straightforward to implement. The basic idea is that two images are subtracted from each other, pixel by pixel. This method is used in land-cover change detection (Sohl, 1999), irrigated crop monitoring (Manavalan et al., 1995), and change detection of forest conversion (Jha & UNNI, 1994). Specifically, the univariate image differencing approach, a type of 2D image differencing, involves calculating the difference between corresponding pixel values from two images taken at distinct time points (Sohl, 1999). This process generates positive, negative, or zero values that signify areas of change or stability. Landsat TM was employed for single-band analysis.

The univariate image differencing method can be represented as:

$$\Delta x = x_1 - x_2 + C \quad (1)$$

where Δx is the change pixel value, x_1 is the pixel value at time 1, x_2 is the pixel value at time 2, and C is a constant.

While it is a relatively simple and straightforward technique, it does not produce detailed change information compared to some more advanced methods. Additionally, it requires manual input for selecting the threshold value, which could introduce subjectivity into the analysis and limit the potential for transfer learning.

2. **Vegetation Index change detection** method functions similarly to image differencing. It involves computing vegetation indices such as the NDVI, PVI, and RVI for each date and subtracting them to identify changes. NDVI is calculated using the formula:

$$NDVI = \frac{NIR - R}{NIR + R} \quad (2)$$

where NIR is the near-infrared band and R is the red band. PVI is derived using the equation:

$$PVI = \frac{NIR - R}{\sqrt{a^2 + 1}} \quad (3)$$

where a is the slope of the soil line in the NIR-R scatterplot. RVI is determined by:

$$RVI = \frac{NIR}{R} \quad (4)$$

By subtracting these indices, the Vegetation Index change detection methods produce detailed change detection information with reduced error impacts from topographic and illumination effects, making it particularly effective for applications such as forest canopy change detection (Nelson, 1983) and land cover (Lyon et al., 1998). A common formula for the Vegetation Index change detection method is:

$$\Delta VI = VI_1 - VI_2 \quad (5)$$

Where ΔVI is the change in the vegetation index, VI_1 is the vegetation index at the first date, and VI_2 is the vegetation index at the second date. However, it should be noted that the Vegetation index change detection method can be sensitive to random noises.

3. **Principal component analysis (PCA)** is a statistical technique used for change detection in multi-spectral remote sensing images. By transforming the original data into a new set of uncorrelated

variables called principal components, PCA effectively reduces data redundancy and utilizes highly correlated multi-temporal data. However, if the input variables are not highly correlated, PCA may not perform optimally in reducing data dimensionality. Additionally, this method requires standardization and relies on expert input for selecting thresholds and key components. Some of the purposes for which PCA has been used are forest defoliation change detection (Muchoney & Haack, 1994), where PCA was used to identify and quantify temporal changes in forest defoliation patterns, and monitoring rapid urban expansion using stacked multi-temporal images (O, 1998). PCA effectively discriminated between built-up and non-built-up areas while highlighting urban expansion patterns and monitoring urban expansion.

4. **Object-based image analysis (OBIA)** is a method that segments images into objects, which are then classified based on their spectral, spatial, and contextual properties. OBIA has been widely applied for change detection in urban areas and forest monitoring (Blaschke, 2010) (Chen et al., 2012). In OBIA, images are segmented using algorithms like the multiresolution segmentation (MRS) that combines pixels into objects based on homogeneity criteria. After segmentation, objects are classified using techniques like the nearest neighbor classifier, support vector machines, or decision trees, which consider the objects' features and spatial relationships. Figure 3 provides a visual representation of the output obtained from the OBIA segmentation process.

However, OBIA's performance is sensitive to the choice of segmentation parameters, and the quality of segmentation directly impacts the accuracy of change detection. Proper parameter selection is crucial to ensure meaningful and accurate object delineation, which can be challenging and often requires expert knowledge.

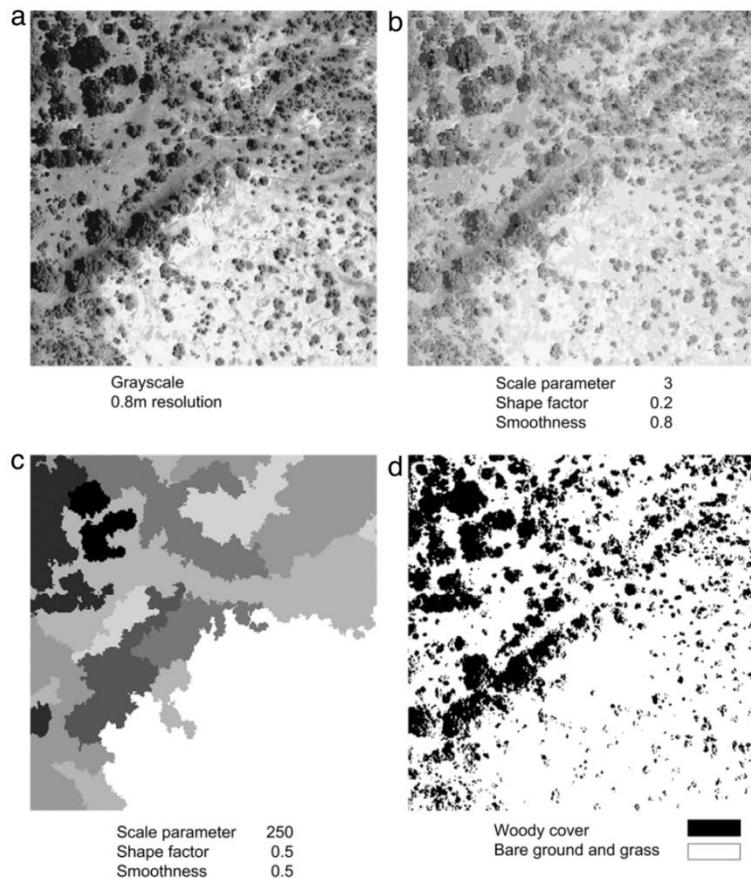


Figure 3 - (a) Aerial photograph of heterogeneous landscape (b) fine-scale segmentation (c) coarse-scale segmentation (d) object-based classification of woody cover, resulting in 97% accuracy

[Image Reference](#)

5. **Random Forest (RF)** is an machine learning method that has been applied to change detection in remote sensing images. RF is based on constructing multiple decision trees, and the final classification is determined by the majority vote of the individual trees(Breiman, 2001). Belgiu and Drăguț (2016) present a comprehensive review of the applications of Random Forest in remote sensing, including its use in change detection tasks (Belgiu & Drăguț, 2016). The authors highlight the advantages of using Random Forest for change detection, such as its ability to handle complex and noisy data, robustness to overfitting, and capacity to provide variable importance measures. They provide examples of studies that have successfully employed Random Forest for change detection in various remote sensing applications, such as land cover change, urban expansion, and deforestation.

Im and Jensen (2005) propose a change detection model that combines neighborhood correlation image analysis with decision tree classification (Figure 4), which can incorporate Random Forest as a classifier (Im & Jensen, 2005). The authors apply their model to detect land cover changes in a rapidly urbanizing area, demonstrating the effectiveness of this method in capturing complex changes.

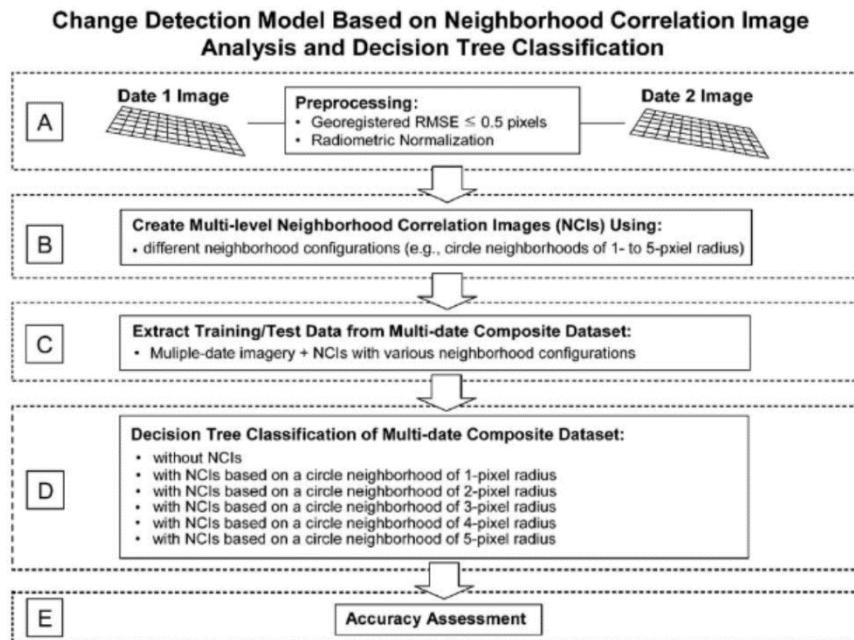


Figure 4 – Flow diagram of the change detection model

[Image Reference](#)

6. **Artificial Neural Network (ANN)** change detection gained popularity and shifted the attention of the majority of researchers (Yotov et al., 2023). ANNs consist of interconnected nodes or neurons, structured in layers, which process and transmit information in a manner similar to the human brain. The ability of ANNs to model complex relationships and learn patterns in data makes them well-suited for change detection tasks in remote sensing. Many researches suppose that ANN is able to outperform all the previous traditional change detection methods. With the new development in technology, the resolution of Remote Sensing data has greatly improved and so have the requirements for accuracy when it comes to change detection. ANN is a supervised method and can predict the data properties based on training samples using an activation function like the sigmoid function: $f(x) = 1 / (1 + e^{-x})$. It performs well in the cases of urban change detection (X. Liu & Jr, 2002) and forest change detection (Woodcock et al., 2001).

Dalwin et al. (2018) proposed a novel feature descriptor for automatic change detection in remote sensing images (Dalwin et al., 2018). The proposed descriptor, aims to provide a more comprehensive representation of the images. This improved representation can then be used in conjunction with an ANN classifier to enhance the change detection performance in various remote sensing applications, such as land cover change, urban expansion, and deforestation (Figure 5).

7. **Convolutional Neural Network (CNN)** change detection is a state-of-the-art method that builds upon the principles of ANNs. Unlike ANNs, only the last layer of a CNN is fully connected, generally resulting in a more powerful network. CNNs are applicable to data with stationarity, locality, and compositionality properties. Convolutional Neural Networks have shown superior performance in 2D image segmentation, including change detection. Siamese CNNs, in particular, have gained attention for their effectiveness in analyzing RGB images for earthquake-induced building damage assessment (Kalantar et al., 2020). When data are significantly different, for example, if images are coming from two types of sensors, weights could be independent, leading to the so-called pseudo-Siamese network (Zhan et al., 2017). Recently, MSCANet (Figure 6), a CNN-based network, has been developed for 2D change detection (M. Liu et al., 2022). This network uses a feature extractor to obtain features from input images and a transformer architecture to capture and aggregate multiscale context information from the features, resulting in better performance than traditional methods.

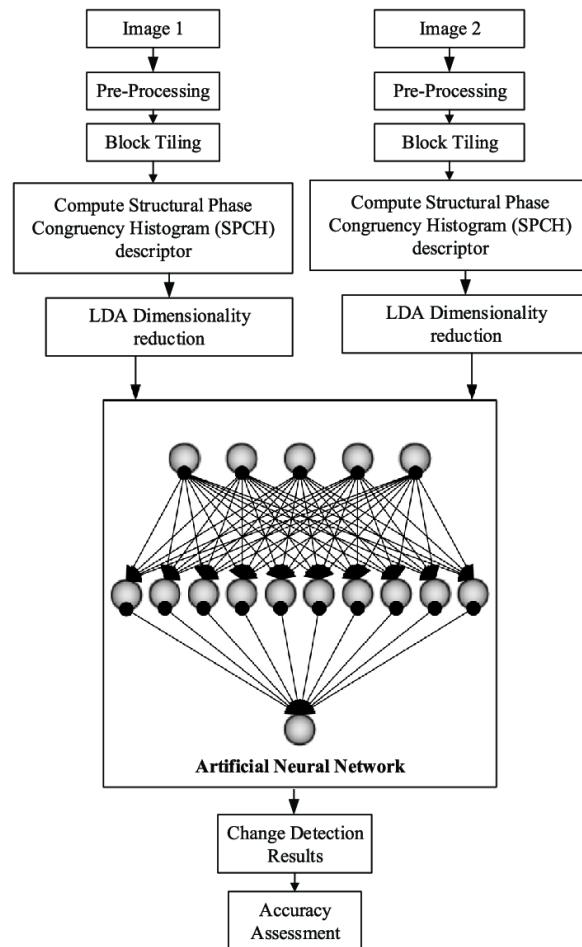


Figure 5 - Change detection using ANN

[Image Reference](#)

2.2. 3D change detection

Real-world problems often involve objects that require height information for accurate change assessment, such as urban areas or terrain changes. In recent years, numerous studies have focused on the development of 3D change detection methods, including those based on point cloud data (Still & Xu, 2023). 3D change detection can be performed using Digital Terrain Models (DTMs), rasterized point clouds, voxel point clouds, and various pre-processed point clouds. Among these techniques, simple methods continue to be popular due to their fast computational speed, ease of implementation, and effectiveness.

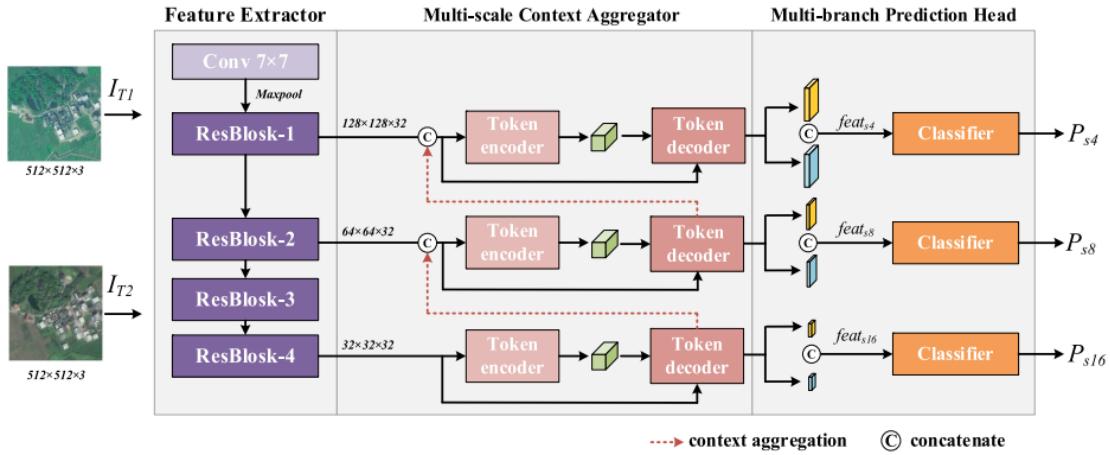


Figure 6 - MSCANet architecture overview

[Image Reference](#)

1. **Octrees** are a common approach for representing and comparing point clouds in 3D change detection. These hierarchical data structures efficiently encode the spatial arrangement of points in 3D space, allowing for fast and robust change detection (Xu et al., 2015). Xu et al. (2015) utilized a point-based method to detect changes in buildings and trees using airborne LiDAR data, where octrees played a crucial role in the process. The study highlighted the importance of data preprocessing, octree generation for non-ground points, and change detection based on the octree structure. As shown in Figure 7, this approach successfully determined changed areas, such as new buildings, changed buildings, demolished buildings, new trees, and removed trees. It distinguished between changes in buildings and trees using a classification and clustering method. This paper underscores the continued relevance and practicality of simple methods, such as octrees, in modern change detection applications.

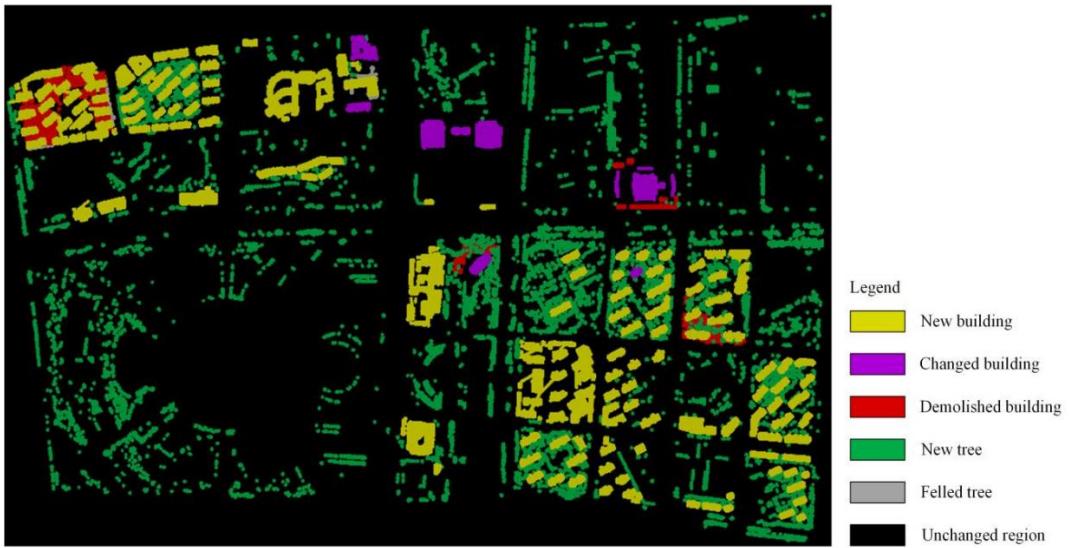


Figure 7 - Change detection results, using Octrees

[Image Reference](#)

2. **The Multiscale Model to Model Cloud Comparison (M3C2) algorithm** (Lague et al., 2013) is another popular technique for comparing point clouds with varying resolutions and densities. It computes distances between point clouds at multiple scales by estimating normal vectors at each point, projecting the points onto their normal vectors, and calculating the mean and standard deviation of the projection distances within a user-defined search radius (Figure 8). This multiscale approach allows for a more robust and accurate measure of distances between point clouds.

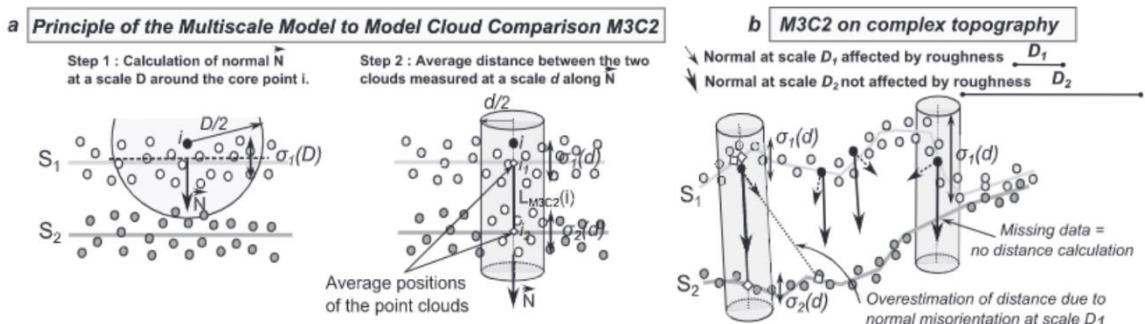


Figure 8 - Overview of M3C2 algorithm

[Image Reference](#)

3. Voxel-based approaches are another class of methods used to compare 3D point clouds. These techniques involve dividing the 3D space into a regular grid of voxels and comparing occupancy or other attributes between the two epochs (K. Liu et al., 2016). Liu et al. (2016) proposed an efficient algorithm for detecting spatial changes using voxel-based representations. Their approach aims to automatically identify differences between two point clouds without making assumptions about their sampling density and distribution. The algorithm constructs a voxel grid, assigns labels to points based on voxel occupancy, and outputs the labeled point clouds. By transforming the point clouds into voxel representations, the algorithm avoids computationally expensive operations like mesh reconstruction and Hausdorff distance computation. The voxel grid also provides a uniform resampling due to its constant voxel size, making the algorithm robust to varying sampling densities. Voxel-based methods are sensitive to the choice of voxel size and may result in information loss

due to the discretization of point clouds. However, they effectively address issues of varying point densities between epochs and occlusion by combining occupancy analysis with additional measurements (Xiao et al., 2012).

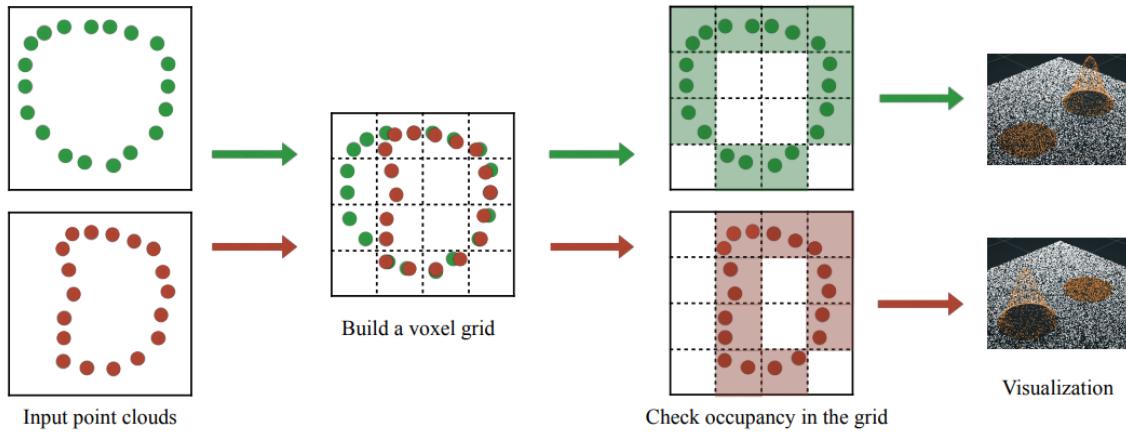


Figure 9 - Comparing the geometry of two point clouds using voxels

[Image Reference](#)

4. Digital Elevation Models (DEMs) and Digital Surface Models (DSMs) are another approach used to compare 3D point clouds by estimating differences based on 2D grid representations. In these methods, height values are compared pixel-wise, and changes are identified using predefined thresholds and pixel occupancy shifts (Stal et al., 2013) (Guerin, Binet, & Deseilligny, 2014). By focusing on height values, DEMs and DSMs provide a simplified representation of the 3D scene that enables efficient comparison and change detection. These methods have demonstrated the effectiveness of using 2D-grid-based difference estimation in DEMs and DSMs for identifying changes between point clouds. Their methodology involves converting the point clouds into raster representations and comparing the height values at each grid cell. Changes are detected by examining the differences in height values and evaluating them against predetermined thresholds. This approach allows for the efficient identification of changes between point clouds while still maintaining an acceptable level of accuracy.

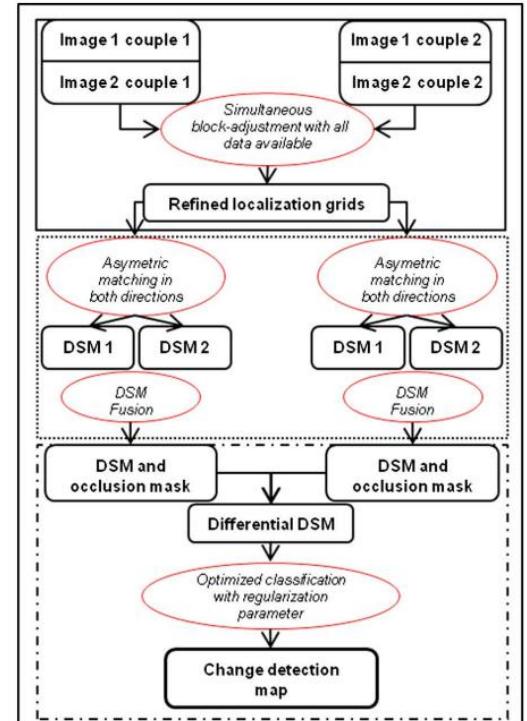


Figure 10 - Workflow of DSM change detection method

[Image Reference](#)

5. Cserép and Lindenbergh (2022) implemented a comprehensive workflow for nation-wide change detection in large airborne laser altimetry point clouds. Their technique, which combines object detection, noise filtering, morphological operations, and clustering, demonstrated a significant improvement in processing efficiency and accuracy, especially in urban areas. Their methodology, applied on the full Dutch altimetry archive, proved that efficient processing and accurate change detection on a national scale is possible with high-performance computing environments (Cserép & Lindenbergh, 2023).

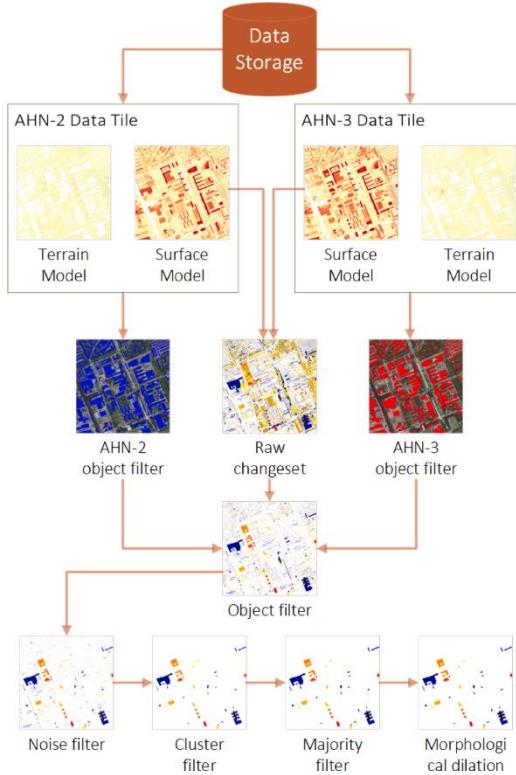


Figure 11 - The complete overview of the algorithm workflow (Cserép and Lindenbergh)

[Image Ref](#)

After discussing various methods for change detection in 3D point clouds, including both traditional and more recent techniques, it is important to consider the application of deep learning approaches. These cutting-edge methods have demonstrated promising results in detecting changes with higher accuracy and efficiency.

Deep learning is a subfield of machine learning that focuses on artificial neural networks with multiple layers, allowing for the automatic extraction and learning of complex patterns and features from raw data. In the context of change detection in 3D point clouds, deep learning approaches have emerged as powerful tools for building higher-level features without the need for user specification, moving beyond traditional distance correspondence in Euclidean space.

Many deep learning approaches for change detection in 3D point clouds involve rasterizing the point clouds into Digital Surface Models (DSMs) or other manageable formats before applying deep learning techniques on the resulting images. However, this gridding process can result in point loss and decreased accuracy in height measurements.

To address these limitations, researchers have developed deep learning methods that can directly process raw point cloud data for change detection. One such method is PointNet (Qi et al., 2017), a deep learning architecture designed to handle unordered point cloud data, enabling the direct classification and segmentation of raw point clouds (Figure 12).

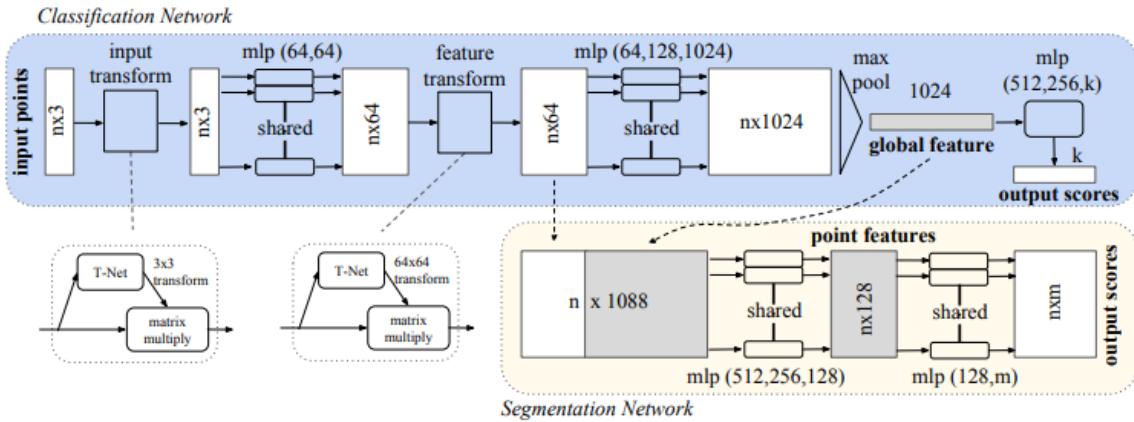


Figure 12 - PointNet Architecture

[Image Reference](#)

(Yastikli & Cetin, 2021) have applied PointNet in their method to classifying raw LiDAR point clouds. By extracting spatial features directly from the point cloud data, the proposed method avoids the need for rasterization or gridding, preserving the accuracy of the original data. The results of the study demonstrate the effectiveness of point-based deep learning methods in handling raw LiDAR point clouds for tasks such as 3D building reconstruction.

In the context of remote sensing, deep learning Siamese networks have proven to be successful and have been widely used (He et al., 2018) (Zhan et al., 2017). A recent study by Iris de Gélis (de Gélis et al., 2023) proposes a deep learning-based method for 3D change detection using raw point clouds. In this study, the authors develop a Siamese architecture that processes and compares point cloud data from two different epochs to identify changes in urban environments. Demonstrating the potential of deep learning techniques for 3D change detection tasks, their approach outperforms traditional methods in terms of accuracy and efficiency. The change detection approach they presented employs the KPConv method, a type of deep learning technique specifically designed for point cloud data. KPConv is a geometric deep learning method based on convolutional kernel point networks that are capable of extracting high-level features from 3D point clouds (Thomas et al., 2019).

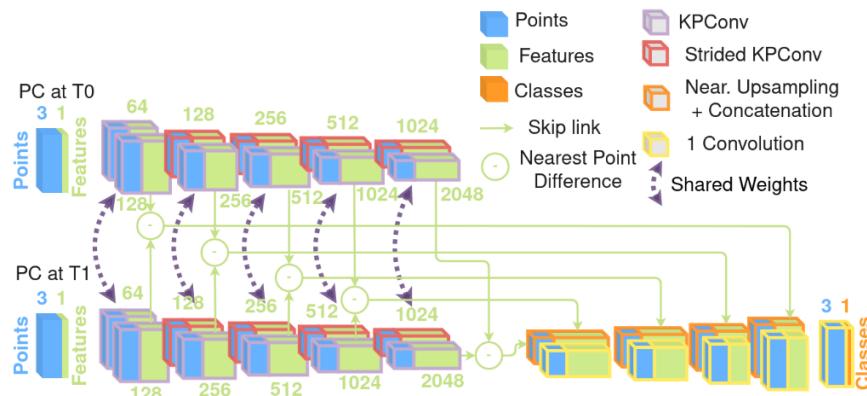


Figure 13 - Siamese KPConv network architecture

[Image Reference](#)

2.3. Research Gap

Change detection in 3D point clouds has become increasingly relevant as remote sensing technologies advance and provide a wealth of spatial data. Although deep learning methods have gained popularity in various fields, including computer vision, speech recognition, and natural language processing(Guo et al., 2019), their application to change detection in raw point cloud data remains challenging.

With the development of deep convolutional networks, image, video, voice, and audio processing have advanced quickly. Furthermore, deep learning has demonstrated incredible outcomes performing tasks related to natural language comprehension (Collobert et al., 2011), classification, sentiment analysis, question answering, and language translation (Jean et al., 2014).

Despite recent advances in automated object detection and segmentation, change detection in raw point cloud data still faces significant obstacles (Longbotham et al., 2012)(Hussain et al., 2013). The challenges arise from the small scale of available datasets, high dimensionality, and the unstructured nature of 3D point clouds. Most of existing deep learning approaches often require pre-processing or data transformation, such as rasterizing point clouds to DSMs, which can lead to a loss of information and reduced accuracy. However, recent efforts like PointNet (Qi et al., 2017), KPConv (Thomas et al., 2019), and the Siamese KPConv (de Gélib et al., 2023) have shown promise in directly processing raw point cloud data. These methods, can be computationally demanding and may require large amounts of training data to achieve optimal performance, making them less suitable for certain applications or situations with limited computational resources.

When semantic segmentation is performed separately on each point cloud, there's a risk of inconsistencies between the two segmentation results. These inconsistencies could be due to variations in data quality, differences in the conditions under which the data was collected, or simply the inherent uncertainty in any semantic segmentation task. It can lead to errors in the differencing step, potentially leading to false positives or false negatives in the change detection results.

Our proposed method seeks to overcome some of these limitations by combining KD-trees and convolutional neural networks in a novel way to process raw point cloud data, aiming for a more efficient and accessible solution for change detection tasks in 3D point clouds. By leveraging the strengths of KD-trees to order and match the data from two epochs, we can efficiently combine features from both epochs extremely fast, despite the high density of the point clouds. Using deep learning networks, we developed an algorithm that performs well while being less computationally demanding and more practical for a wider range of applications. The process of training and predicting the models uses concepts that have been explored and proven successfully in 2D change detection. In this study we explore three different models :

1. **RF algorithm**
2. **Fully Connected Neural network**
3. **Convolutional Neural Network**

This strategy not only takes advantage of the strengths of deep learning methods but also addresses the unique challenges associated with 3D point cloud data. Our methodology, which directly compares raw point clouds from two epochs using deep learning models, has the potential to capture subtle changes that might be missed with segmentation techniques.

Our research goal is to develop a deep learning algorithm that overcomes the limitations of existing methods, is fast, efficient and opens a new avenues for research in the geoinformation science field.

3. DATASETS

This study uses two different point cloud datasets to experiment with our algorithms. The first dataset, known as Actueel Hoogtebestand Nederland (AHN), offers comprehensive and exact elevation information spanning across the Netherlands, making it an ideal real-world dataset. The second dataset used is [Urb3DCD](#), a simulated 3D city, useful for understanding urban transformations. By using two different datasets, we gained a deeper understanding of the characteristics of 3D data which helped us to determine the most effective ways to utilize them for our research. Also, it was interesting to compare how our algorithms will perform in a real versus a simulated scenario.

3.1. AHN dataset

The AHN data can be freely accessed, downloaded, and used. We obtained this data through the GeoTiles.nl website ([Geotile Source](#)). This website is a valuable resource that offers color-coded AHN point cloud data available in smaller tiles. This tiling system has already proven its worth for TU Delft and is now available to everyone. Tiles from the AHN series (AHN 1, 2, 3, and 4) come in a size of 1x1.25 km, making them perfect for processing, viewing, and breaking down into smaller tiles as inputs for our algorithms.

Depending on the area, the actual time difference between the AHN2 and AHN3 is between four and ten years. The AHN4, collected between 2020 and 2022, offers a faster-paced update for the entire Netherlands. Our study primarily uses AHN3 and AHN4 data, further elaborated in Table 1 and Table 2. Figure 14 illustrates the acquisition plan for both AHN3 and AHN4.

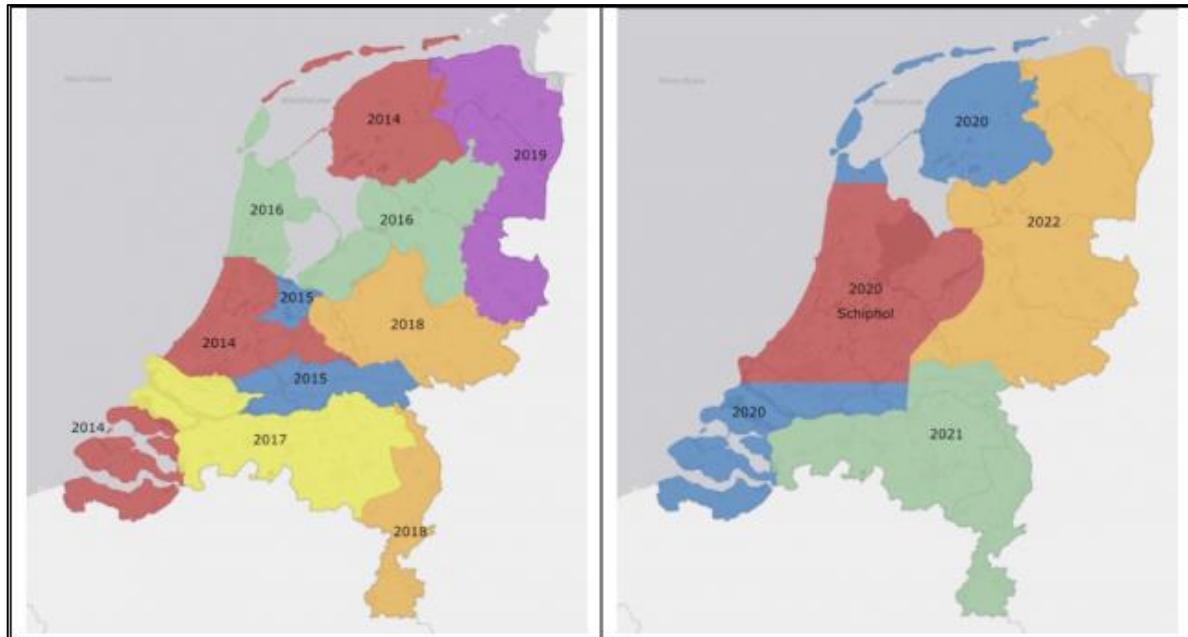


Figure 14 - Data acquisition years for AHN3 data (left) and planned years of data acquisition for AHN4 (right)

Table 1 - AHN dataset properties

| Dataset characteristics | | | | |
|--------------------------------|---|--|----------------------------|---|
| | Point density | Classes | Year of acquisition | Labels we used |
| AHN 3 | Average between 6 and 10 points per square meter | ground level (ground), buildings, works of art, water and other. | 2014-2019 | x, y, z, red, blue, green, intensity, number of returns |
| AHN 4 | 10-14 points per square meter. The area around Schiphol is 20-24 points per square meter. | ground level (ground), buildings (limited), works of art (limited), water and other. | 2020-2022 | |

Table 2 - AHN dataset accuracies

| Dataset accuracy | Height accuracy | Planimetric accuracy |
|--|------------------------|-----------------------------|
| Systematic error | 5 cm | 8 cm |
| Stochastic error | 5 cm | 5 cm |
| At least 68.2% of the points have a height accuracy of: | $5 + 1 * 5 = 10$ cm | $8 + 1 * 5 = 13$ cm |
| At least 99.7% of the points have a height accuracy of: | $5 + 3 * 5 = 20$ cm | $8 + 3 * 5 = 23$ cm |

The location of our areas chosen for this study are shown in the Figure 15 below:

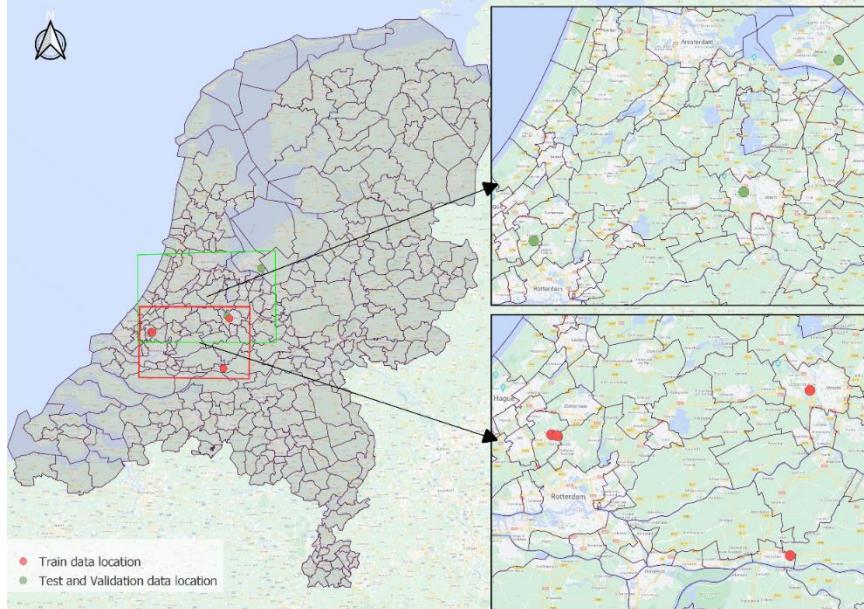


Figure 15 - AHN dataset locations

The major distinction between the datasets, as shown in Table 1 and Table 2, is their point density and acquisition year. Having said that, this dataset is ideal for our research since we don't have to worry about fluctuating accuracy, which might introduce uncertainties in identifying changes.

The AHN dataset was collected under time constraints, which inevitably influenced its final quality. We were developing a new concept and testing it on two different datasets, which limited our ability to gather a larger dataset. While we strived to ensure the data was as complete and accurate as possible, it's worth noting that it may not match the level of detail and balance found in the dataset used by Iris de Gelis and her team, with which we make most of our comparisons with. Despite these differences, our AHN dataset still provides a solid foundation for our research, although it's important to consider these factors when comparing our results with those derived from more curated datasets.

3.2. URB3DCD dataset

Urb3DCD is an open-access 3D point cloud simulated dataset (de Gélis et al., 2021). It consists of multi-epoch pairs of point clouds generated through an urban point clouds simulator, which models aerial LiDAR surveys over urban areas. This dataset has seven classes of changes available, such as unchanged, new building, demolition, new vegetation, vegetation growth, vegetation loss, and mobile objects. There isn't any information about the thresholds of "vegetation growth" in the paper by Iris de Gelis or the website, but it appears to involve a significant change in the z value. Also, since this is a simulated dataset, "unchanged" means there's absolutely no change in height. This would mean that vegetation classified as "unchanged" keeps its original height. To verify this, we performed a visual inspection of the data ourselves. Changes in the urban landscape, such as building construction or demolition, are introduced by the simulator and directly annotated at the point level. Some of the properties of this data are shown in the Table 3 below:

Table 3 - Urb3DCD dataset properties

| Urb3dCD Lidar dataset | |
|----------------------------------|-----------|
| Density (points/m ²) | 0.5 |
| Noise range across track (0) | 0.01 |
| Noise range along track (0) | 0 |
| Noise scan direction (m) | 0.05 |
| Scan angle (0) | -20 to 20 |
| Overlapping (%) | 10 |
| Height of flight (m) | 700 |
| Annotation level | Point |

The dataset accurately represents the complexity of real-world urban environments, including variations in building designs, vegetation, and other urban features. The point cloud pairs represent the same area at two different points in time. The location of the simulated dataset is Lyon, France and the total area used for training is 3.6 km², or 10 tiles of 0.36 km².

However, the accuracy of building data labels is not always perfect, which, in a sense, makes it a suitable training set for our algorithms. Real-world scenarios rarely present us with immaculately annotated data. At times, the indicated changes on the simulated dataset are not entirely precise, as they reflect a general change zone rather than the exact delineated shape of the object. The problem arises when this kind of labelling persists in the test and validation data (Figure 16) causing uncertainties in our accuracy assessment.

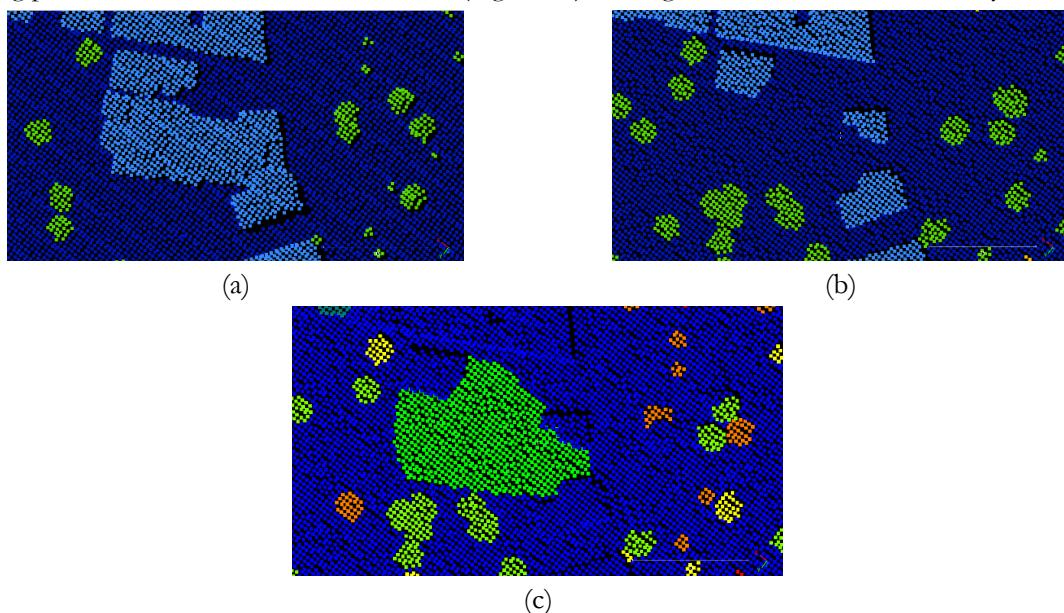


Figure 16 - Noise with the labeling of Urb3DCD data. The change (c) happening from AHN3 (a) to AHN4 (b) is not delineated correctly. In (a) and (b), light blue represents the buildings, green is the vegetation and blue is the ground. In (c) we see the shape of the building change in colour green which doesn't represent the correct shape for the actual change.

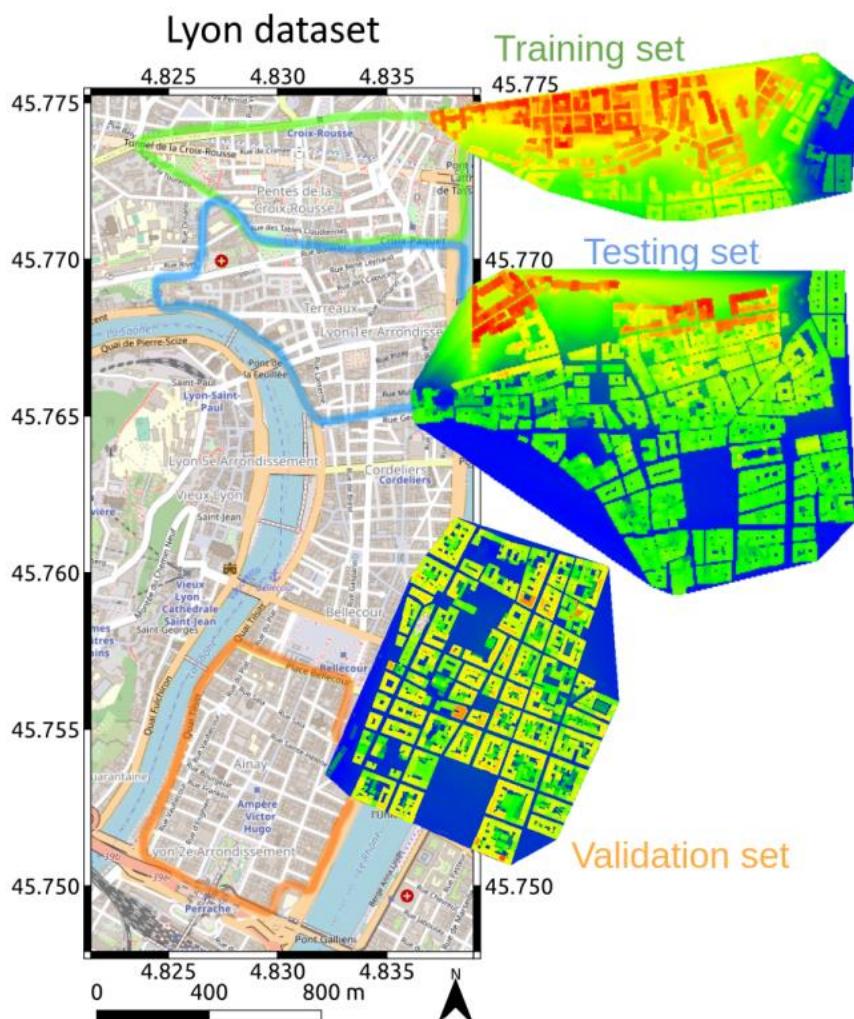


Figure 18 - The location of the simulated datasets (Lyon, France)

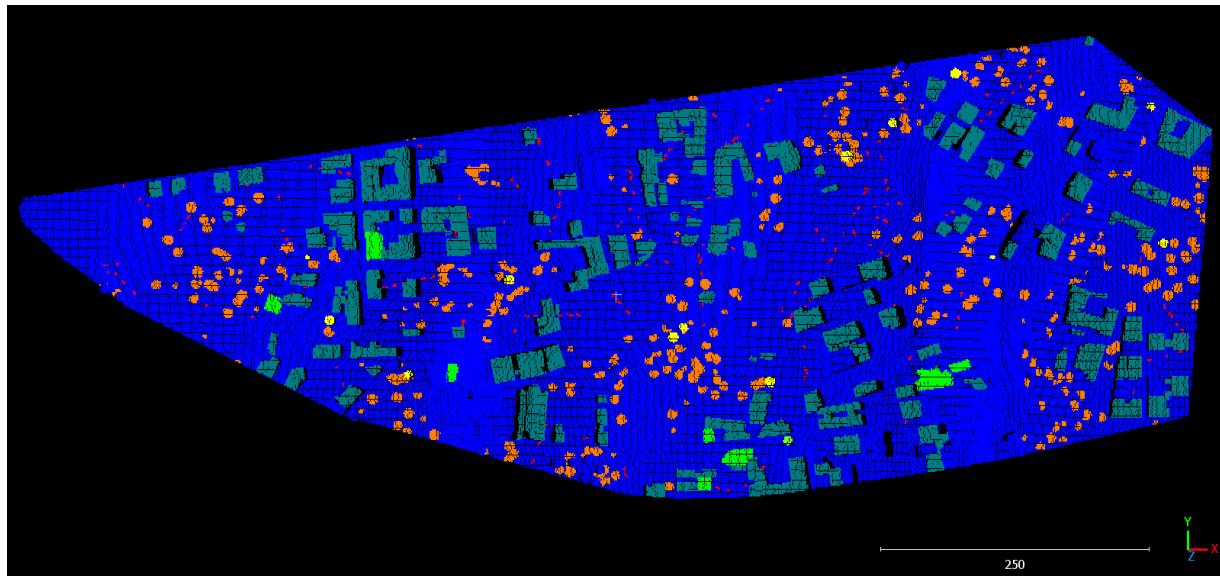


Figure 17 - One of the training tiles of the Urb3DCD dataset

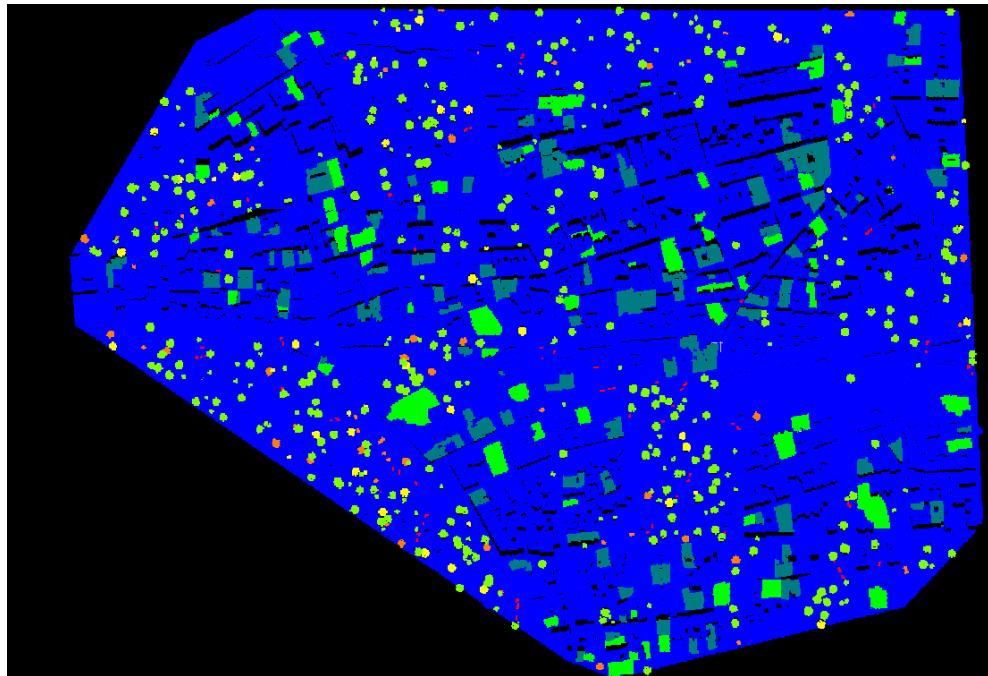


Figure 20 - One of the testing tiles of the Urb3DCD dataset

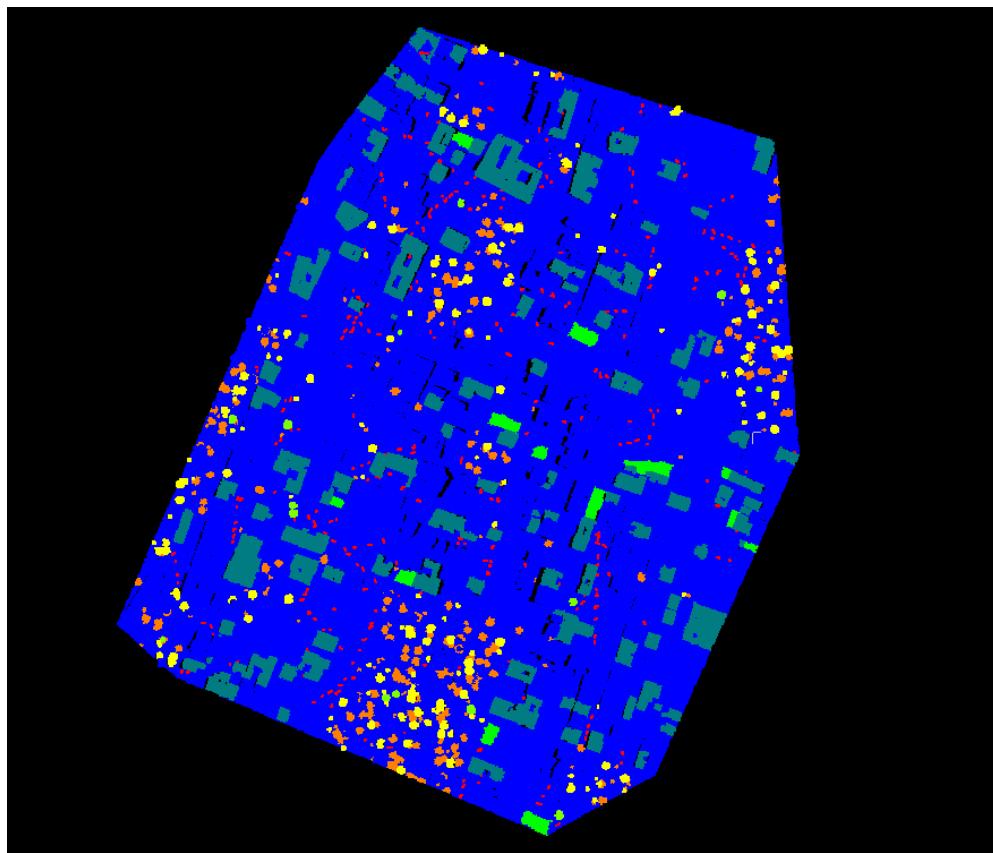


Figure 19 - One of the validation tiles of the Urb3DCD dataset

In summary, the Urb3dCD dataset provides researchers with a valuable set of training, testing and validation data that can be used to develop 3D point cloud change detection algorithms in urban settings.

4. METHODOLOGY AND ALGORITHMS

The aim of this study is to detect changes in urban areas using labeled point clouds. This process includes: pre-processing of datasets, the preparation of data for the developed models, and the development of networks for change detection techniques. The change detection and classification of point clouds is a complex process, and the considerations and motivations for our methods are detailed in the following sections.

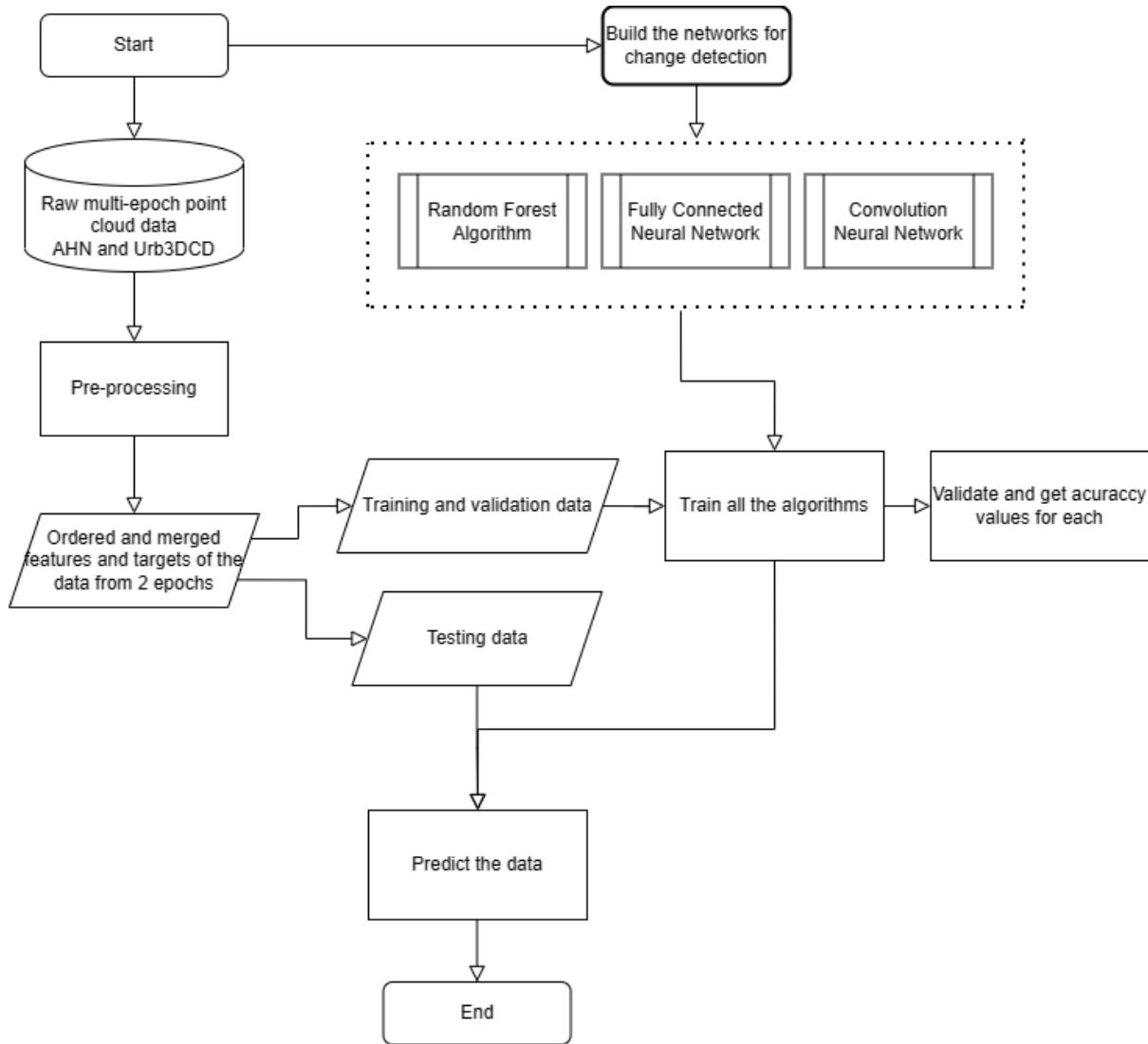


Figure 21 - An overview of the methodology

4.1. Concept and Approach

Our approach starts with two different labeled point cloud datasets: the Actueel Hoogtebestand Nederland (AHN) and Urb3DCD datasets. The AHN datasets consist of points labeled as urban elements (building, water, ground, etc.), which serve as the basis for generating training data that represents 'changed' and 'unchanged' urban areas. Each point in both these datasets is a multi-dimensional feature space, which includes x, y, and z coordinates, and additional color attributes such as red, green, and blue. In addition, the AHN data includes intensity and number of returns. Figure 22 shows the same area across two epochs, AHN3 and AHN4, labeled with the classes mentioned in "Datasets" section. These tiles are our base data for the generation of the training, testing, and validation sets.

The AHN and Urb3DCD datasets require different pre-processing steps due to their inherent properties. The Urb3DCD dataset, being a balanced representation of 'changed' and 'unchanged' points while also having them labeled and ready for training, requires minimal pre-processing. On the other hand, the AHN dataset demands a more hands-on approach and more preprocessing steps due to its very dense nature, the imbalance between 'changed' and 'unchanged' points, and the absence of changed labels.

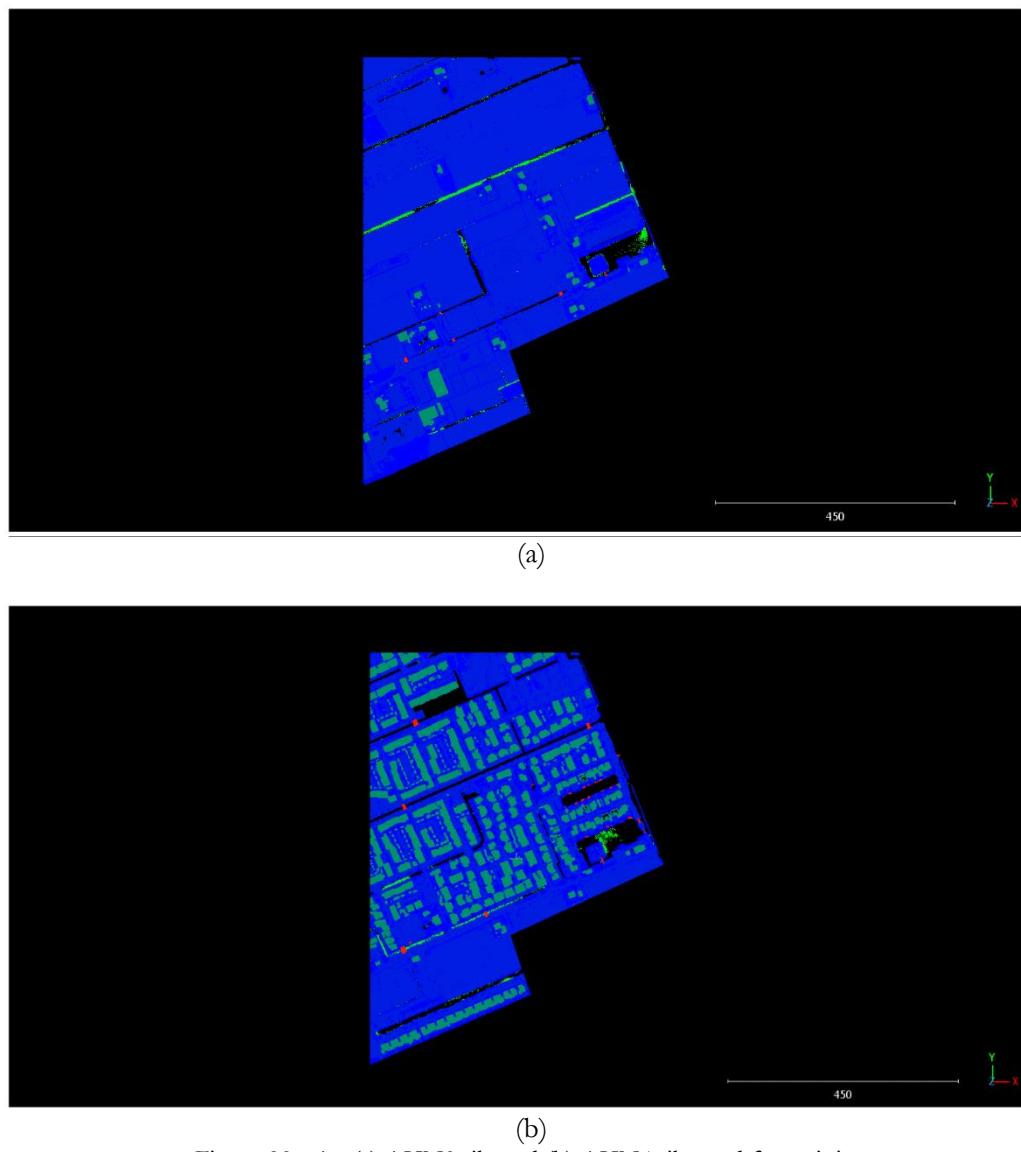


Figure 22 - An (a) AHN3 tile and (b) AHN4 tile used for training

After getting the data (Figure 22), our process starts by selecting the nearest point or multiple of the nearest points between two epochs. In simpler terms, we overlap the tiles from 2 different epochs and match

the nearest points with each other. The whole process is done via a Python code, which is available in our Github Repository. We adopt this strategy of selecting the nearest point for several reasons:

1. **Spatial Relationship Consideration:** To begin with, point cloud data contain complex spatial relationships that require careful considerations. Selecting the nearest point(s) from one epoch to the other takes into account the spatial context, identifying corresponding locations between two epochs based on their proximity. It is the simplest and most intuitive way of mapping changes between two datasets.
2. **Ensuring Uniform Distribution between epochs:** The data from two epochs have uneven numbers of points per area, because of point density differences, and the data was collected at different times. By pairing points from two epochs one-to-one and discarding unmatched points, we resolve this issue and ensure a fairly equal distribution of points per area between two epochs.
3. **Data Compatibility for Algorithms:** Nearest point selection provides a logical and efficient way to combine features from two different point cloud sources into a single input array for our algorithms. Machine learning models, such as Random Forest (RF), and deep learning models such as Neural Network (NN), and Convolutional Neural Network (CNN), require input data in a specific format, usually an array of data points and a corresponding array of targets. We make sure that our data is properly formatted for the model input, by pairing the closest points and combining their features.
4. **Integration of Temporal Changes:** It allows us to integrate the attributes of the nearest points from two distinct datasets, creating a new set of data that captures the changes in urban areas over time. Each type of change (new building, new vegetation, etc.) in our dataset will have a unique combination of features. Thus combining the features of points from the same area in different epochs can lead to an accurate method of detecting changes.
5. **Generating labels for training:** By matching the nearest points, we can generate change labels, for example, if the point in epoch 1 was a building and the nearest point to this point in epoch 2 is another class, we label it as “demolition”. This is a fairly accurate and extremely fast method of creating training and validating data. Even though this method is not perfect, since we are only generating “unchanged”, “new building” and “demolition” classes, is really easy for us to conduct visual inspection of the training, testing and validation data before we feed them into the algorithm. Thus, the generated data will be of a really high quality when it comes to labeling, so no issues are expected to come from this process.

4.2. Generation of Training Data

The next step is to generate training, testing, and validation data from the Nearest Neighbor-matched datasets. This process includes labeling points as different classes of 'changed' and 'unchanged,' which serve as inputs for the change detection models. While the Urb3DCD dataset already contains 7 unique change labels, for the AHN dataset, we will be using the Pandas Data Frame in Python to generate these labels ourselves. This library is very efficient in working with large tabular data.

The point clouds from both AHN epochs are uploaded in python and converted to two Pandas Data Frames. Then we combine column-wise into a single Data Frame. To generate the target labels for our analysis, we apply a custom function on the new Data Frame. This function creates a “change label” column, which assigns a label based on the changes in the classification information between the two epochs. For example, if the label was “building” on the first epoch and then it’s something else on the second epoch, it will be labeled as “demolition”. Figure 23 shows the general logic of how the change labels are generated and Figure 24 shows how this data looks after we labelled it.

Generating balanced training data is one of the challenges of this methodology. While the Urb3DCD dataset provides a somewhat balanced representation, the AHN dataset requires careful preparation. The urban regions in the Netherlands experienced minimal changes during the interval between the collection of AHN3 and AHN4 data, thus the data is unbalanced, with a higher number of 'unchanged' points. Due to the big size of the AHN dataset, it is necessary to manually ensure that there is a good balance

between 'changed' and 'unchanged' points. This imbalance can be addressed through undersampling 'unchanged' points, oversampling 'changed' points, or manually cropping the tiles for balanced classes.

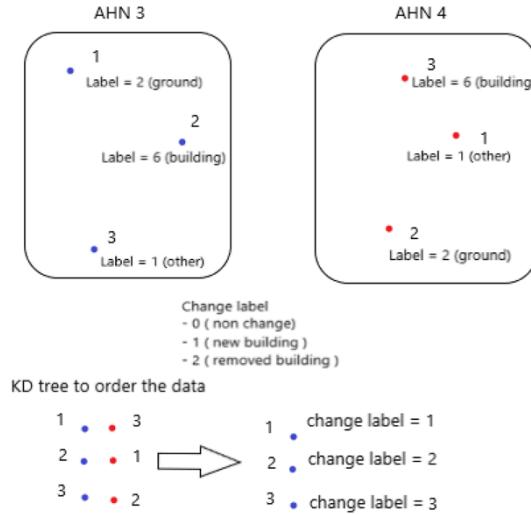


Figure 23 - Target Label Generation

We choose to proceed by manually cropping our tiles in order to ensure that classes are equally represented. This is the fairest way of data representation and keeps the data realistic. The data balance helps in avoiding biases in the learning algorithms and improves the accuracy of the change detection results. The AHN tiles were selected and cropped manually and then downsampled, reducing the size of the data and making it more manageable for processing and analysis. As seen from the example in Figure 24, we manually cropped the area to ensure that the number of points in each class is somewhat the same. This process not only reduced the data size to make the process computationally possible but also ensured a reasonable representation of the urban environment, creating a balanced dataset that mirrors the change dynamics in an urban setting.

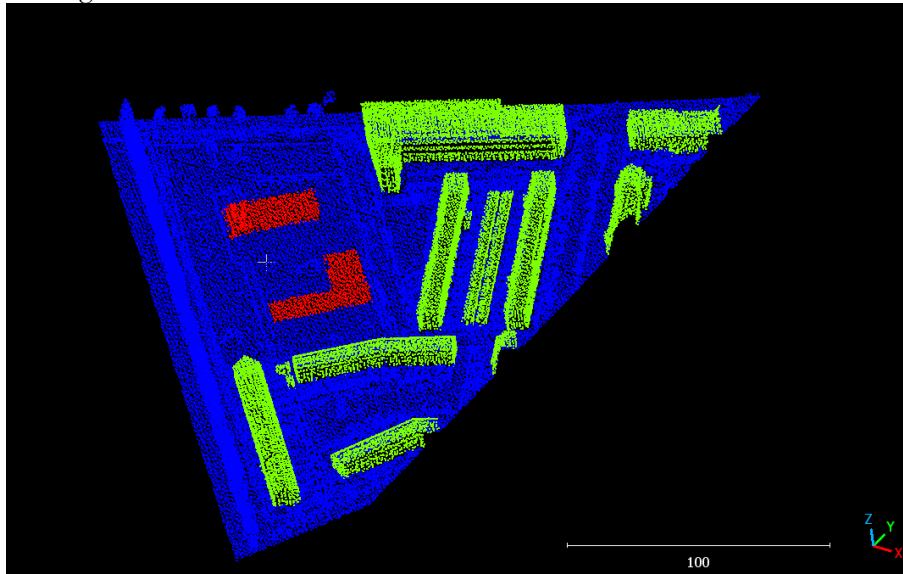


Figure 24 - A balanced AHN tile used for training. Red is for new buildings, Green is for removed buildings and blue is unchanged.

A change cannot be represented by a singular point or a small number of point clouds, especially in the context of very dense point clouds such as AHN. But there still are many small groups of points, wrongly labeled as changes after we generate the target labels. To address this issue, we apply a clustering

algorithm to eliminate small outliers mislabelled as changes. Figure 25 and Figure 26 show how this clustering helps in getting better training data.

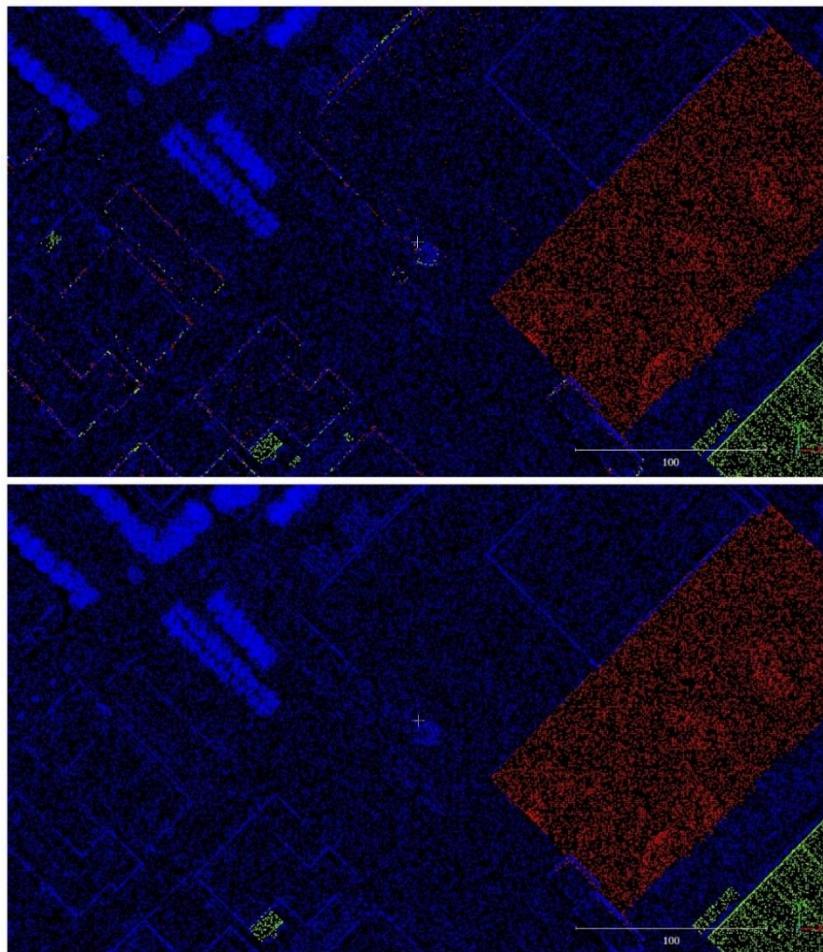


Figure 25 - A closer look into the data cleaning. Original (up) and cleaned (down)

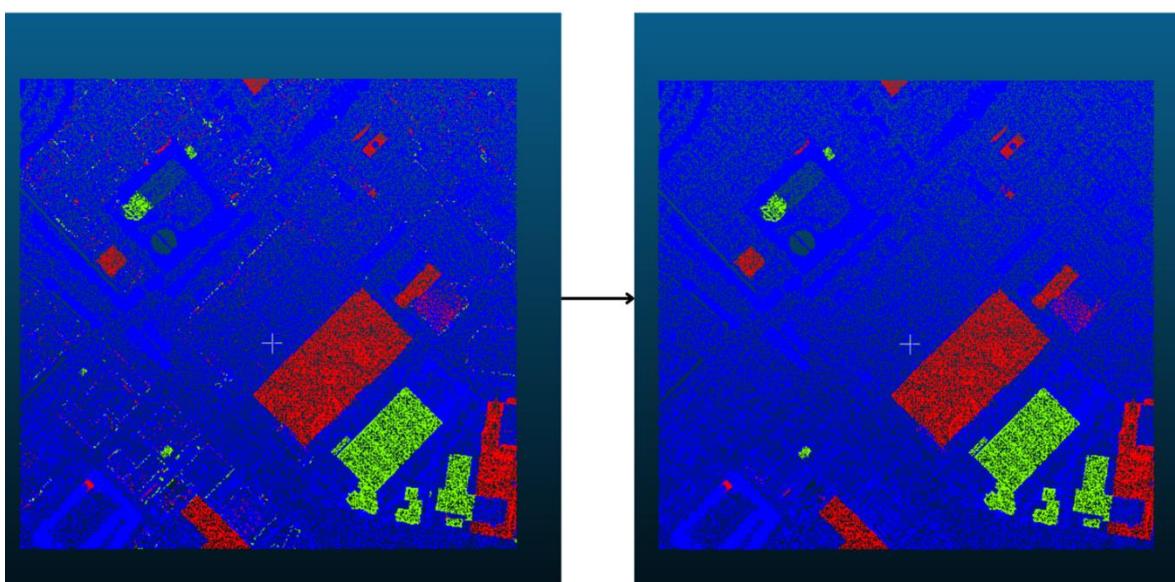


Figure 26 - AHN training tile before (left) and after (right) the point cleaning

During the process of preparing our data for change detection with the AHN dataset, we observed that a significant portion of the changes were part of the class "new buildings", while instances of "demolition" were comparatively very low. To address this imbalance and ensure an equal representation of all classes in our training labels, we choose to swap the epochs of some tiles. Meaning that for some tiles we used AHN4 as our first epoch and AHN3 as our second epoch, and this way we diversified our dataset and balanced the number of points in different categories.

Before we jump to the models it is important to discuss the features used in AHN dataset. As mentioned above, from the original source we get Z, RGB, Intensity and Return Number. Now if we look at Figure 27 (a) and (b), notice that even though there were no buildings present in AHN3, the RGB values still indicated that. This kind of error is present very often, and during training process we noticed that it really lowers our algorithms accuracy and leads to many wrong predictions. Even though RGB as features have proven to be really valuable on classification and change detection tasks, we cannot use them for this study. In order to add one more feature and somewhat implement a feeling of spatial surrounding to our AHN data, we manually crafted the feature called "Point density on 2D plane". This feature is the total amount of points around each point in a 2D radius of 0.8m. The idea behind it is that the point density of non-planar objects like trees will be much higher than planar objects like ground or rooftops. This feature is expected to increase the performance of our non-spatial models.

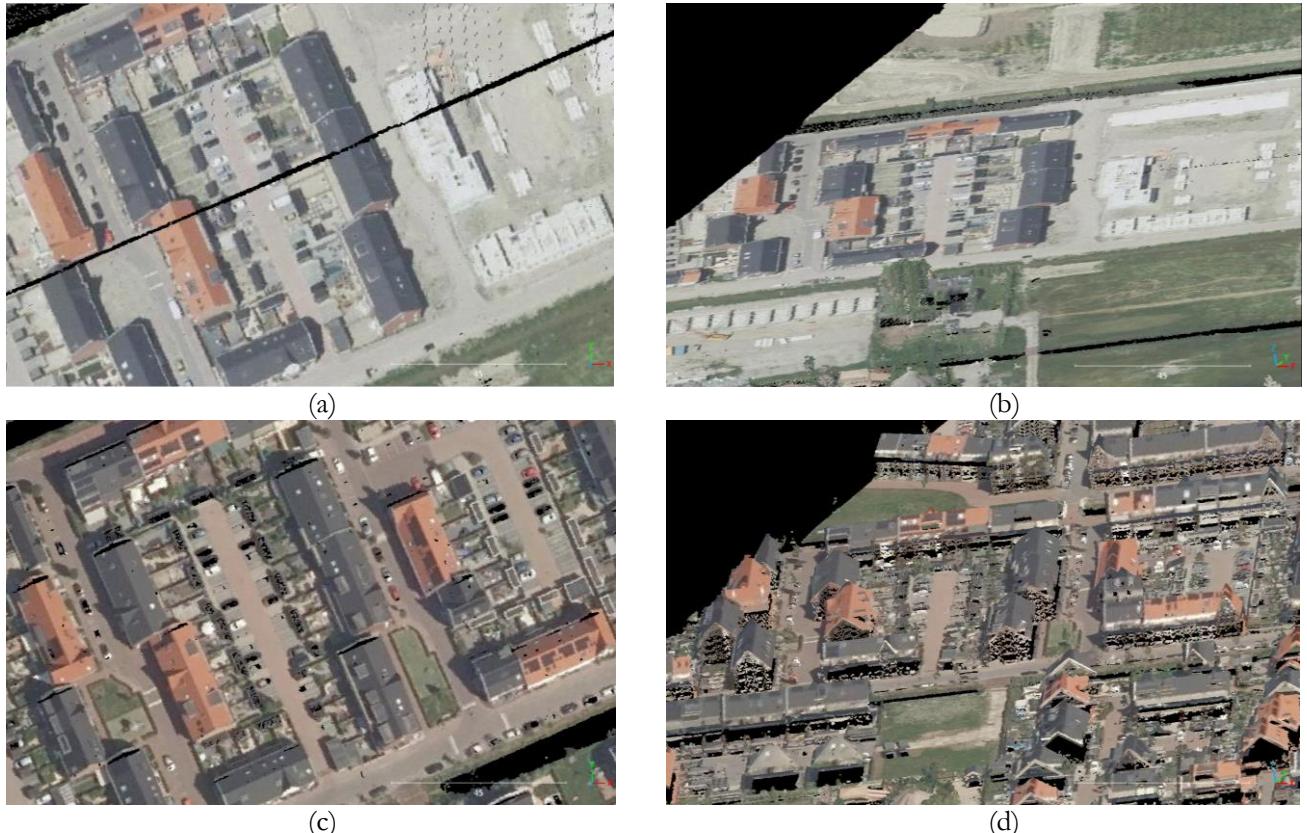


Figure 27 - AHN tiles, where (a) and (b) are the same area in the AHN3 with RGB from different points of view. (c) and (d) are also the same area but in AHN4

Before feeding the data into the model, we apply feature scaling using the Standard Scaler from the sklearn pre-processing module. This step standardizes the data by transforming it so that it has a mean of 0 and a standard deviation of 1. Data scaling helps the algorithm converge faster and improves its performance, as now all our features are on a comparable scale.

Now that our data is ready, we have to choose and design our change detection methods.

4.3. Change detection models

In this study, we use three change detection methods: Random Forest (RF), Fully Convolutional Neural Networks (FCNN), and Convolutional Neural Networks (CNN). Each method was selected for its unique strengths and effectiveness in dealing with large data sets, and because they have demonstrated good performance in similar tasks. By choosing RF, FCNN, and CNN, we want to experiment with different approaches, while also comparing these models under the same conditions. Our method of detecting changes is not only applicable to the datasets used in this study but can also be adapted to different datasets and urban environments, making it very flexible for urban change detection.

The Random Forest was chosen due to its robust performance in handling large datasets, high-dimensional data, and its ability to deal with the noise and missing data, common in point clouds. We expect it to be the fastest model, providing consistent results across various data scenarios. Furthermore, the RF algorithm is more interpretable as compared to the deep learning models since it allows the calculation of feature importance scores which could give additional insights into the learning process of the algorithm. The Fully Connected Neural Network, on the other hand, excels in generating complex hierarchical representations and thus was selected for its potential in capturing the data structures within our dataset, particularly in instances of complex urban changes. Finally, the Convolutional Neural Network was included for its superior ability in handling data with spatial properties, making it the optimal choice for our change detection purpose. Together, these models provide a comprehensive approach for change detection in urban environments: RF for reliability, FCNN for complexity, and CNN for incorporating spatial structures.

The sub-sections below will start with an overview of the Random Forest algorithm, followed by an explanation of the Fully Connected Neural Network, and finally, we will talk about our most effective model, the Convolutional Neural Network.

4.3.1. Change Detection using Random Forest Classifier

Random Forest (RF) is a machine learning algorithm known for its strong performance in classification tasks and the ability to handle large amounts of data with high dimensionality (Breiman, 2001). Because our data is per-point classified and obtained via an aerial laser scanner, it contains random noise and missing information. RF performs really well even with noisy or missing data, making it suitable for our purpose. To improve the overall prediction accuracy and avoid overfitting, the random forest classifier combines the output of multiple decision trees. Furthermore, RF can estimate the importance of each feature and provides a balance between speed and accuracy, making it a versatile choice for a wide range of applications. Due to the classification nature of our data, we considered using and testing this algorithm for our study.

The Random Forest classifier used in this study is trained on the pre-processed point cloud data. It analyses the pre-processed features extracted from the raw point cloud, such as the height, colour, return number, and intensity of the points (depending on the dataset we use). These features are used by the individual decision trees to make decisions leading to a final classification for each individual point cloud (Figure 28). The Random Forest algorithm does not take into account the spatial relationships between points nor considers their neighbours, thus the classification process is based only on individual points. The goal of this model is to use it compare how spatial and non-spatial models perform. We could try and add features like 2D point density (that we added on the AHN data training) to somewhat represent a special relation between points, but it wouldn't be as effective as we'd want. Because this method of introducing "a spatial feeling" to the model is less refined compared to CNN models which take all features of nearby points into account. So, we'd essentially be inventing only some of the features that the CNN already better handles in a more advanced way. Ultimately we will be trying to make a straightforward and simplistic model such as RF, more complex to just try and reach a small percentage of what CNN already does.

For this algorithm, a set of hyperparameters were tested through a grid search. The grid search involves testing multiple combinations of hyperparameters to identify the best-performing configuration. Specifically, we experimented with different values for the number of estimators (50, 100, and 200) and maximum tree depth (5, 10, and 20). This resulted in a total of 9 combinations. Once the optimal hyperparameters were identified, the classifier was trained on the dataset and used for change detection and classification tasks. The trained classifier was then validated on the validation point cloud dataset and used to predict on a different test dataset.

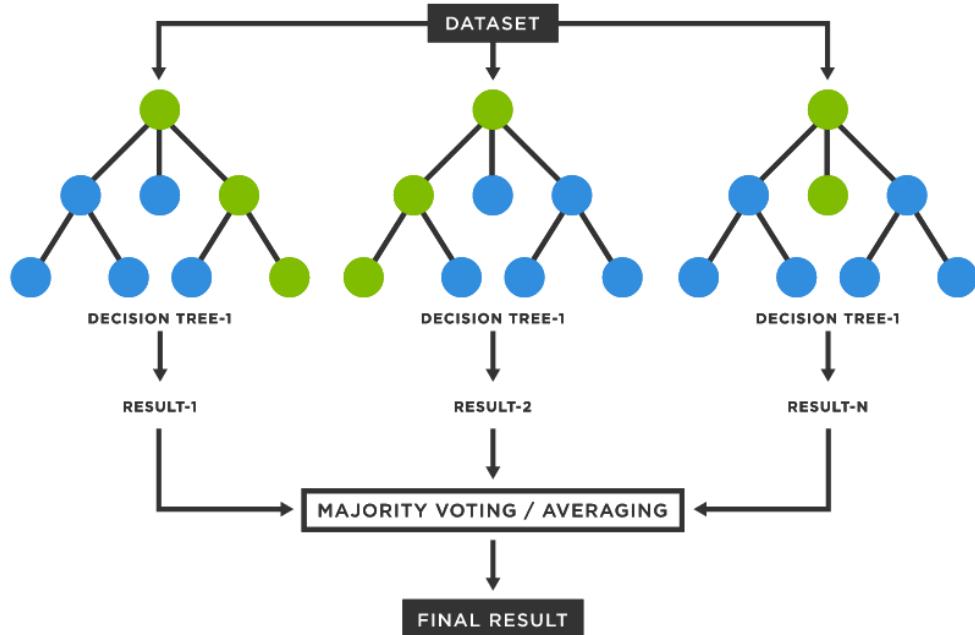


Figure 28 - The model of a Random Forest algorithm used for classification

The entire process was implemented in Python, and our code is freely accessible for use on our GitHub. The code uses sklearn library, which offers a user-friendly interface for training and using the model. The results, accuracies and comparison with the other methods are shown in the “Results and Analysis” and “Discussion” sections.

4.3.2. Change Detection using Fully Connected Neural Network

Random Forests can handle complicated datasets and perform well, but they are unable to automatically extract higher-level features from our lower-level inputs. The ability of Fully Connected Neural Networks (FCNNs) to capture complex representations, in contrast, makes them superior. FCNNs are deep learning algorithms, where the input features are initially passed through a series of hidden layers. Each layer has a set of neurons and each neuron in a layer is connected to all the neurons in the previous layer, forming a fully connected network (Goodfellow et al., 2016). The output of each neuron in a layer is a linear combination of the inputs it receives, followed by a non-linear activation function. Thus through these hidden layers FCNN combines the low-level features and creates high level, more complex features.

FCNNs can accept input of any size and generate output of corresponding dimensions, making them ideal for handling large-scale data. This makes them particularly effective for large urban environments.

In this study, FCNNs were used as a point-to-point methods for change detection, without taking any spatial relation or neighbourhood point into consideration. Our FCNN is designed to work with the inputs generated from the pre-processed data, which includes features and unique classes for each type of urban change. The FCNN architecture was designed, trained and implemented using the Keras library with

TensorFlow as the backend. The model was trained using the same training dataset, and the performance was compared with the other methods to identify the most suitable architecture. In the following paragraph, we will discuss the specifics of our FCNN architecture :

Our FCNN architecture was designed as a deep network and consists of four hidden layers with 256, 128, 64, and 32 neurons. It has an output layer with 7 neurons, one for each class of change. The FCNN architecture uses LeakyReLU and PReLU activation functions in the first two hidden layers, with ReLU activations in the remaining hidden layers. The output layer makes use of the softmax activation function for multi-class classification. The model also incorporates batch normalization and dropout layers to improve generalization. It is compiled using the Adam optimizer and as the loss function we used is sparse categorical cross-entropy. These layers were chosen to extract features from the input data, learn non-linear relationships between the features, and classify the data into target classes. The specific architecture of the model was designed based on common practices in deep learning and an attempt to balance model complexity with the risk of overfitting. The full model as shown in Figure 29 represents a visualisation of the layers used and an overall idea of how our code looks like. In each row of the figure we show three main elements, starting from the left we show the layer name, the type of process per layer (input or output) and the number of neurons this layer uses.

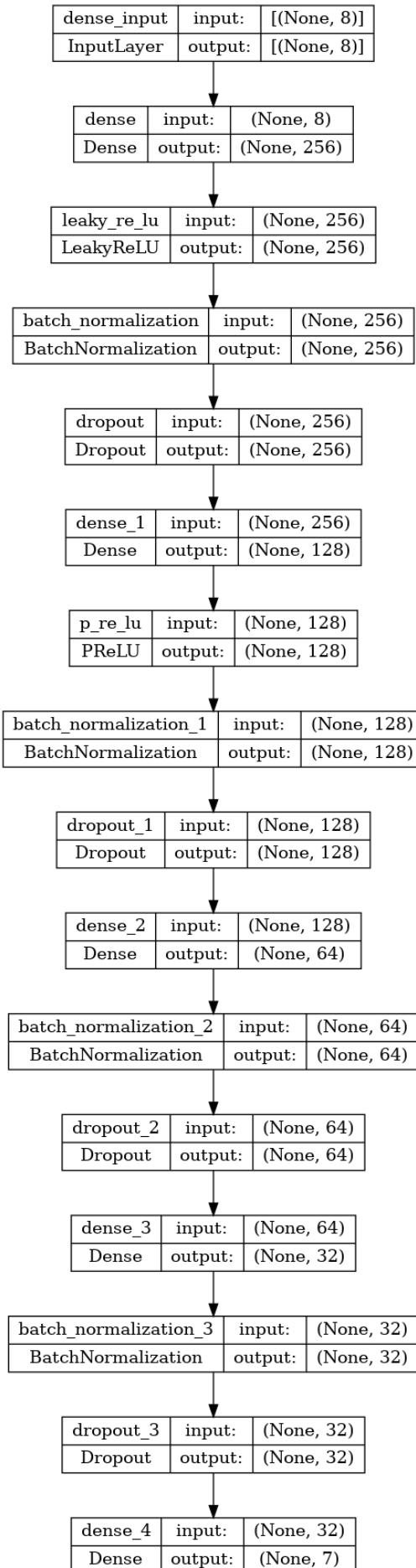


Figure 29 - Our FCNN model

4.3.3. Change Detection using Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of Neural Networks designed to work and process with data that have spatial properties, such as images or point clouds. This type of network applies local filters called convolutions to capture spatial relationships among neighbouring data (Goodfellow et al., 2016). This feature of CNNs is especially helpful with point cloud data, which contains spatial structures. CNNs leverage the spatial context of point clouds, allowing for more effective extraction and learning of complex patterns. Two Convolutional Neural Network (CNN) architectures were designed and trained for change detection in our point cloud data.

Our network was designed using the Sequential model from the Keras library. The CNN architecture incorporates Batch Normalization and MaxPooling1D layers to improve the model's performance. The model has three sets of Conv1D layers with 128, 64, and 32 filters, each followed by Batch Normalization, another Conv1D layer, and a Dropout layer with a rate of 0.3. The dropout rate is applied between the Dense layers to prevent overfitting. MaxPooling1D layers are added after the first and third sets of Conv1D layers. After the Conv1D layers, the model contains a Flatten layer and three Dense layers with 128, 64, and 32 units, each followed by Batch Normalization and Dropout layers. The final Dense layer has 7 units and the output layers uses a SoftMax activation. The architecture is designed to ensure a balance between the model complexity and the ability to learn abstract features from the data.

In our work, we investigated different neighbouring points numbers for the CNN models, using 5, 10, and 15 neighbours in Urb3DCD dataset and 10, 20, 50 and 100 neighbours in the AHN dataset. The reason for doing so is the huge difference in these datasets density. The algorithms were executed with two different batch sizes, 32 and 64, and tested across 8, 10 and 15 epochs.

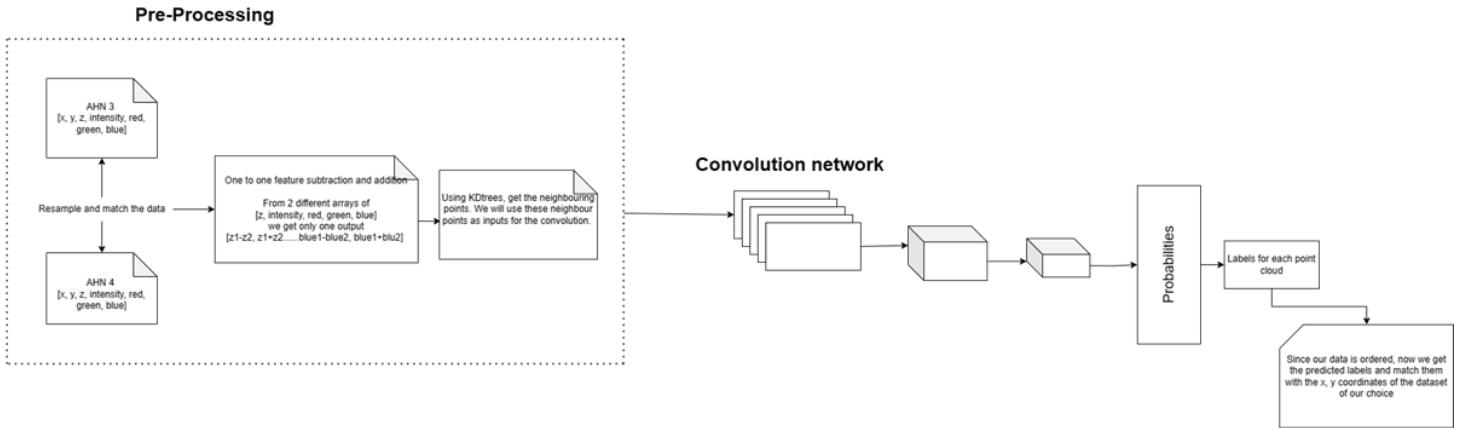


Figure 30 - A summary of the process used for the CNN models

The diagram above offers a comprehensive overview of our discussions thus far and outlines our proposed way forward, specifically focusing on our primary model, the CNN. Previous methods have attempted to add features with each other, and recently they tried subtraction (de Gélis et al., 2023). These efforts are logical as they aim to integrate features from both epochs, which works well for changes that can be described linearly, like variations in point height across two epochs. The subtraction and sum of the height feature of a pair of points that didn't change, will be very different from the case of non-change. This logic can be extended to all features, as they are represented as real numbers. Operations such as multiplication and division would be redundant, as the changes we observe in feature numbers are linear, and these operations would not contribute additional meaningful information.

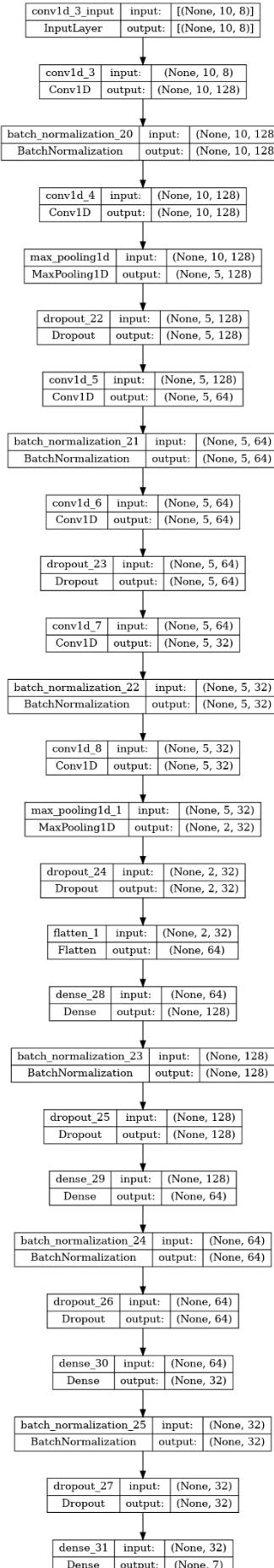


Figure 31 - Our CNN model

4.4. Experimental Design and Parameters

This section talks about the experiments conducted using different datasets and their parameters.

In the context of the Urb3DCD dataset, accurate feature representation, and balanced distribution of changes allowed us to focus exclusively on fine-tuning models parameters. With regards to Random Forest (RF), we experimented with hyperparameters by using a grid search on RF to pinpoint the optimal parameters. For the Deep Learning (DL) models, we adjusted epochs, batch sizes and experimented with a huge variety of layer combinations. For Convolutional Neural Networks (CNN) we also experimented with the number of neighbours. Given that the Urb3DCD data only includes height and RGB information, there was no room to experiment with feature combinations. For DL models, we tested batch sizes of 32 and 64 over 8, 10, and 15 epochs. In the case of CNN training, we also experimented with 5, 10, and 20 neighbours.

In contrast, the AHN dataset offered substantial potential for more experimentation. Similar to Urb3DCD, we conducted a grid search for RF to determine the best hyperparameters. This dataset features height, RGB, Intensity, and return number, allowing us to attempt many feature combinations. Unfortunately as mentioned above, it was not viable for us to make use of RGB values, and this will for sure lower the model expected results as RGB values are really important in classification and change detection tasks. We implemented an alternative approach for RF and Neural Networks (NN) by introducing new feature ourselves, the number of points within a 2D buffer for each point. This feature was crafted due to the lack of utility of RGB values for our AHN model, so an additional feature was needed. Given that the point density is significantly greater than the Urb3DCD dataset, we had to expand the number of nearest points considered in the CNN, experimenting with 20, 50, and 100 points. For the DL models, we experimented with batch sizes of 32 and 64 over 10, 15, and 30 epochs.

In the next section, we will only discuss our highest-performing models, and share our insights and deliberations on them. With that said, our top performers that we will discuss are:

For Urb3DCD: We utilized all available features (z and RGB). The Random Forest was tuned with a max depth of 20 and a number of estimators set at 200. DL models were trained with a batch size of 64 over 10 epochs, and for the CNN, we chose to use 10 nearest neighbours.

For Ahn: We chose to make use of only z, intensity, and return number as our features. The RGB values were wrong in most of the cases, thus the use of them lowered our accuracy. The Random Forest was trained with similar hyperparameters as Urb3DCD - a max depth of 20 and 200 estimators. DL models were trained with a batch size of 64 over 10 epochs, and for the CNN, we used 50 nearest neighbours.

For both: We add and subtract the features with each other (so z_1-z_2 , z_1+z_2 , $\text{intensity}_1-\text{intensity}_2$, etc), then before inputting them in the models we use a normalization code to make sure all values of all features are on the same range (from zero to one). All the basic features used are the default values that are available on the original datasets.

5. RESULTS AND ANALYSIS

To properly validate each method, we trained them three times in the simulated dataset: once with the complete train dataset and then with each half the train dataset. This approach intends to assess the change detection methods' adaptability in response to various quantities of training data. Since we lack data in the real dataset, there we only experiment with the full dataset. In this section we will show results and talk only about the numerical statistics while in the “Discussion” section we will analyse everything in details. Explanation of the terms used in the tables on the coming sections :

Accuracy: the percentage of correctly classified samples out of the total number of samples.

Precision: the ratio of true positives to the total number of positive predictions.

Recall: the ratio of true positives to the total number of actual positive samples.

F1 score: the harmonic mean of precision and recall.

Support: the number of samples in each class in the test set.

IoU : Intersection over Union is the ratio of True Positives (TP) to the sum of True Positives, False Positives (FP), and False Negatives (FN).

5.1. Results for URB3DCD Dataset

In this section, we will talk about the results obtained from our three change detection methods: Random Forest (RF), Fully Connected Neural Network (FCNN), and Convolutional Neural Network (CNN), applied on the Urb3DCD dataset. The full training dataset consists of 1,638,678 points with the following distribution of class labels :

Table 4 - Urb3DCD classes distribution

| Class Names | Class Number | Points in the first half dataset | Points in the second half dataset | Points in the full dataset | Percent proportion (Full dataset) | Percent proportion (First Half dataset) | Percent proportion (Second Half dataset) |
|-------------------|--------------|----------------------------------|-----------------------------------|----------------------------|-----------------------------------|---|--|
| Unchanged | 0 | 672922 | 685754 | 1376971 | 83.57 | 84.39 | 84.03 |
| New Building | 1 | 47198 | 41978 | 90472 | 5.86 | 5.17 | 5.52 |
| Demolition | 2 | 50288 | 50686 | 101580 | 6.25 | 6.24 | 6.2 |
| New Vegetation | 3 | 7896 | 12901 | 21389 | 0.98 | 1.59 | 1.31 |
| Vegetation Growth | 4 | 4288 | 1479 | 5891 | 0.53 | 0.18 | 0.36 |
| Vegetation Loss | 5 | 18760 | 12846 | 31721 | 2.33 | 1.58 | 1.94 |
| Mobile Objects | 6 | 3867 | 6919 | 10654 | 0.48 | 0.85 | 0.65 |
| Total | | 805219 | 812563 | 1638678 | | | |

5.1.1. Random Forest Classifier

a. Full dataset trained random forest

Table 5 - Accuracies of RF model trained with the full dataset

| Class | Class number | Precision | Recall | F1 Score | IoU | Support |
|-------------------|--------------|-----------|--------|----------|-------|---------|
| unchanged | 0 | 0.969 | 0.958 | 0.964 | 0.930 | 403061 |
| new_building | 1 | 0.811 | 0.931 | 0.867 | 0.765 | 27266 |
| demolition | 2 | 0.848 | 0.863 | 0.855 | 0.747 | 43722 |
| new_vegetation | 3 | 0.371 | 0.926 | 0.53 | 0.360 | 2908 |
| vegetation_growth | 4 | 0.817 | 0.454 | 0.584 | 0.412 | 9063 |
| vegetation_loss | 5 | 0.81 | 0.733 | 0.77 | 0.630 | 7306 |
| mobile_objects | 6 | 0.981 | 1 | 0.99 | 0.981 | 2698 |
| accuracy | | | | 0.936 | | 496024 |
| macro avg | | 0.8 | 0.84 | 0.79 | | 496024 |
| weighted avg | | 0.936 | 0.936 | 0.936 | | 496024 |

The model achieved an overall accuracy of 93.59% on the full test dataset. The precision, recall, and F1 score for most classes were satisfying. Our random forest model has a good balance between false positives and false negatives. Class 0 (unchanged) had the highest support and showed the best results with a precision of 96.9%, recall of 95.8%, and F1 score of 96.4%. Class 1 (new building) and class 2 (demolition) also showed good results, with F1 scores of 86.7% and 85.5%, respectively. However, class 3 (new vegetation) had the lowest F1 score of 53.0%, despite having a high recall of 92.6%. The low precision means that it frequently misclassifies this class, while the high recall means that the RF model is able to identify most of the instances that do belong to this class.

Class 4 (vegetation growth) and class 5 (vegetation loss) showed a moderate F1 scores of 58.4% and 77.0%, respectively. Class 6 (mobile objects) had the highest precision and recall, both at 100%, resulting in an F1 score of 99.0%.

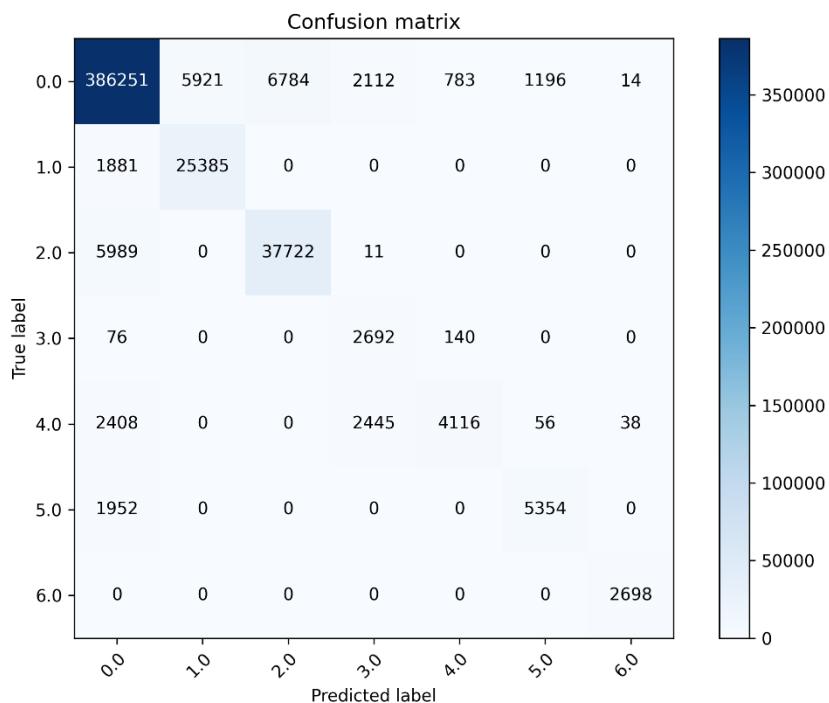


Figure 32 - Confusion matrix of RF model trained on the full dataset

The macro-averaged precision, recall, and F1 score provide an overall assessment of the model's performance across all classes, weighting each class equally. The macro-averaged metrics are all lower than the weighted averages, thus the model performs better on the bigger classes. The confusion matrix (Figure 32), provides a visual representation of the model's performance on each class.

b. Half dataset trained Random Forest classifier

All the accuracies and confusion matrixes for both of the half datasets are available on the "Appendix" section.

For the first half of the dataset, the Random Forest model showed an overall accuracy of 91.6%. The performance for each class, in terms of precision, recall, and F1-score, was generally good, although there were some disparities. The 'New Vegetation' class showed a relatively low precision but a very high recall.

The Random Forest model's performance on the second half of the dataset was slightly higher than the first half, with an overall accuracy of 92%. As in the first half, the 'New Vegetation' class had a low precision but high recall. The 'Vegetation Growth' class showed a significantly low recall of 0.18, indicating a high number of false negatives.

Random Forest Analysis

The model's accuracy decreased from 93.6% to 92% and 91.6% when we experimented with it in the half datasets. This reduction in accuracy suggests that the model relied on a larger quantity of data to learn the underlying patterns. The impact that the reduction of the training data caused, was not uniform across all classes. The analysis on the paragraphs below is done between the full dataset and the worst performing half dataset model.

Class 2 (demolition) experienced a considerable drop in its recall (from 86.3% to 62.2%), leading to a decrease in its F1 score from 85.5% to 72.1%. This indicates that the model had difficulty identifying true positives for this class when we trained it on a smaller dataset. As a straightforward model, the Random forest heavily relies on input features quality and diversity as it cannot create more complex features out of them. This was a very good example to show this limitation. It is really important to train in very similar data on what we want to predict.

Class 4 (vegetation growth) saw a decline in its F1 score from 58.4% to 55.9%. On the other hand, class 6 (mobile objects), maintained perfect precision and recall even after we reduced the training data. This implies that the model could still learn the patterns associated with this class, despite having less data to work with. Meaning that random forest models don't need a lot of training data, as long as the data is diverse enough to cover everything we have to predict.

We observed that the model's performance in predicting class 3 (new vegetation) was relatively consistent, as shown by similar F1 scores for both the full and half datasets. This suggests that even with a reduced amount of data, the training set was sufficiently representative to capture all potential feature combinations that the model might encounter in the validation dataset. The model struggled with certain classes, particularly 'New Vegetation' and 'Vegetation Growth', which had low precision and recall, respectively. The model's struggle to accurately classify new vegetation might not be solely due to the size of the training data, but could also be attributed to other factors such as feature selection or model parameters.

In summary, the Random Forest Classifier, has shown considerable strengths, particularly in its ability to handle large datasets and its robustness to overfitting. The impact of reduction of the training data was not the same across the classes. While some classes experienced decline in performance, others remained stable or were not affected. This analysis highlights the importance of the selected features, having a sufficient amount of training data and the data should be representative enough so it covers all possibilities we might encounter in the validation and testing dataset.

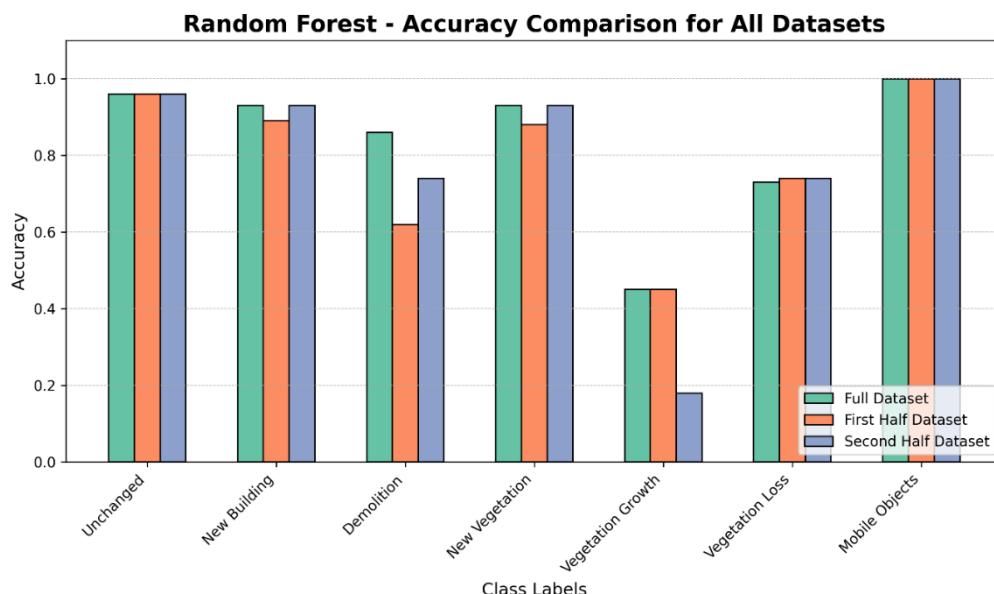


Figure 33 - Per-class accuracy of all datasets for the RF model, Urb3DCD

5.1.2. Fully Connected Neural Network

a. Full dataset trained FCNN

Table 6 - Accuracies of FCNN model trained with the full dataset

| Class | Class number | Precision | Recall | F1-score | IoU | Support |
|-------------------|--------------|-----------|--------|----------|-------|---------|
| unchanged | 0 | 0.972 | 0.957 | 0.964 | 0.931 | 403061 |
| new_building | 1 | 0.81 | 0.962 | 0.88 | 0.785 | 27266 |
| demolition | 2 | 0.846 | 0.868 | 0.857 | 0.750 | 43722 |
| new_vegetation | 3 | 0.371 | 0.926 | 0.53 | 0.360 | 2908 |
| vegetation_growth | 4 | 0.829 | 0.438 | 0.573 | 0.401 | 9063 |
| vegetation_loss | 5 | 0.789 | 0.764 | 0.776 | 0.634 | 7306 |
| mobile_objects | 6 | 1 | 1 | 1 | 1 | 2698 |
| accuracy | | | | 0.937 | | 496024 |
| macro avg | | 0.802 | 0.845 | 0.797 | | 496024 |
| weighted avg | | 0.943 | 0.937 | 0.938 | | 496024 |

The overall accuracy of the model is 93.7% on the full dataset. The results are as below:

1. Unchanged (class 0): With a precision of 97.2%, a recall of 95.7% and an F1-score of 96.4%, the model showed very good performance in predicting this class. Based on the precision value for this class, when a point is predicted to be 'unchanged', it is correct 97.2% of the time. The model is successful in identifying 95.7% of the actual 'unchanged' instances.

2. New building (class 1): While the precision of 81% is relatively good, it is the recall value of 96.2% that stands out for this class. High recall means that the model is highly sensitive to the 'new building' instances and successfully identifies the majority of them. However, the relatively lower precision indicates the presence of false positives, with samples from other groups being incorrectly classified as 'new building'. This difference in precision and recall results in an F1 score of 88%, indicating the importance of balance.

3. Demolition (class 2): The models performance on this class has precision of 84.6%, a recall of 86.8%, and an F1-score of 85.7%. In this case we have a balance between precision and recall. These results show that FCNN is able to predict 'demolition' points in most cases.

4. New_vegetation (class 3): This class shows the most significant differences between precision and recall. A low precision of 37.1% and a very high recall of 92.6% means that there is a significant number of false positives present during the prediction. Our FCNN model is overestimating the 'new vegetation' class, thus it is misclassifying points that belong to other classes as this class. This imbalance is reflected in the F1-score of 53%, suggesting that there is considerable room for improvement.

5. Vegetation_growth (class 4): The model has a high precision (82.9%) but low recall (43.8%) for this class. This suggests that while the model's predictions for 'vegetation growth' are usually accurate, it fails to identify a big portion of actual 'vegetation growth' points.

6. Vegetation_loss (class 5): The precision of 78.9% and recall of 76.4% indicate a fair balance, leading to an F1-score of 77.6%. These numbers indicate that the model's performance is relatively consistent.

7. Mobile_objects (class 6): The model performs flawlessly for this class, achieving perfect precision, recall, and F1-score.

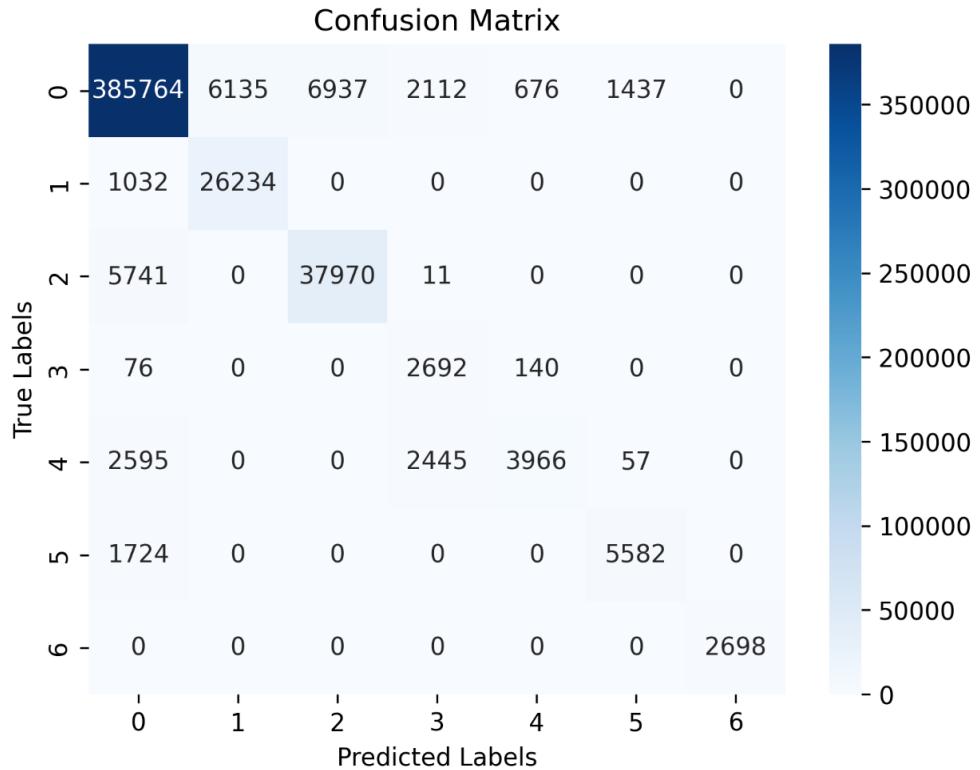


Figure 34 - Confusion matrix of FCNN model trained on the full dataset

b. Half dataset trained FCNN

All the accuracies and confusion matrixes for both of the half datasets are available on the “Appendix” section.

Our FCNN model that was trained on the first half of the dataset, demonstrated an overall high accuracy of 93.6%. Precision, recall, and F1-score for each class were also generally high. The model performed well for this data subset. However, it is important to note the variations across different classes. The 'New Vegetation' class, despite its high recall of 0.926, had a low precision of 0.371, indicating a high rate of false positives. 'Vegetation Growth' also showed a similar trend, with a high precision of 0.84 but a low recall of 0.421, indicating a high number of false negatives. These imbalances may be attributed to the uneven distribution of class instances in the dataset.

The model's performance on the second half of the dataset was slightly lower, with an overall accuracy of 93%. The precision, recall, and F1-score for most classes changed only by a small percentage. Similar to the results from the first half dataset, the 'New Vegetation' class had a low precision but high recall, while 'Vegetation Growth' showed high precision but low recall.

Neural Network Analysis

The FCNN model performed very well across all classes, with an overall accuracy above 93% in all three experiments. The model demonstrated very high performance for the 'Unchanged' and 'Mobile Objects' classes. Even when trained on different subsets of the data, the model consistently achieved high accuracy, indicating its ability to generalize well.

The best results were achieved by class 0 (Unchanged). This class has the highest support value, thus the model's high performance on this class could be heavily influenced by it. However, the results also highlight areas for potential improvement. In particular, the low precision for the 'New Vegetation' class and the low recall for the 'Vegetation Growth' class shows that the model struggles to differentiate vegetation-related classes. Looking at the results, the 'Vegetation Growth' class is particularly challenging to classify, possibly due to overlapping features with other vegetation classes. The difference on performance between the model trained on the full dataset and those trained on the first and second halves was minimal. This indicates that our model is robust to variations in the data quantity.

Nevertheless, the imbalances in class distribution across the datasets, impacted the model's performance for some classes. Future work could explore techniques such as class balancing, better class representation and oversampling of minority classes to improve performance across all classes.

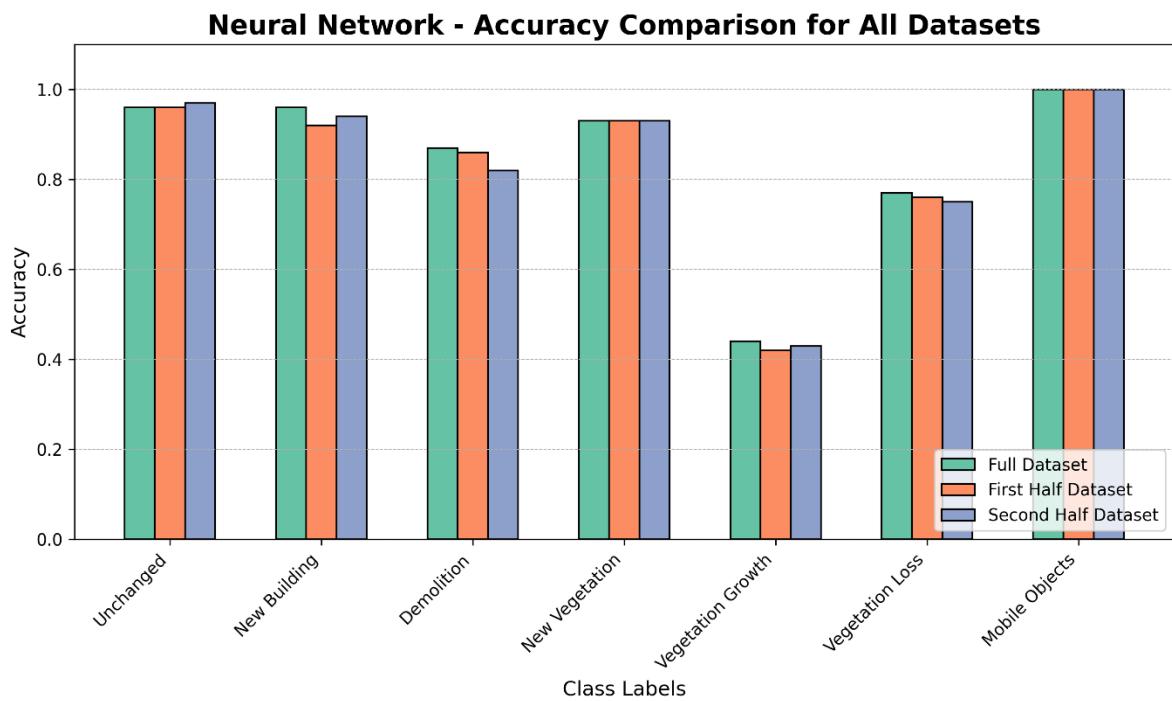


Figure 35 - Per-class accuracy of all datasets for the FCNN model, Urb3DCD

5.1.3. Convolutional Neural Network

a. Full dataset trained CNN

Table 7 - Accuracies of CNN model trained with the full dataset

| Class | Class number | Precision | Recall | F1-score | IoU | Support |
|-------------------|--------------|-----------|--------|----------|-------|---------|
| unchanged | 0 | 0.968 | 0.971 | 0.97 | 0.941 | 403061 |
| new_building | 1 | 0.871 | 0.924 | 0.897 | 0.813 | 27266 |
| demolition | 2 | 0.954 | 0.808 | 0.875 | 0.778 | 43722 |
| new_vegetation | 3 | 0.372 | 0.925 | 0.53 | 0.361 | 2908 |
| vegetation_growth | 4 | 0.684 | 0.661 | 0.672 | 0.506 | 9063 |
| vegetation_loss | 5 | 0.781 | 0.768 | 0.774 | 0.631 | 7306 |
| mobile_objects | 6 | 1 | 1 | 1 | 0.999 | 2698 |
| accuracy | | | | 0.946 | | 496024 |
| macro avg | | 0.804 | 0.87 | 0.817 | | 496024 |
| weighted avg | | 0.95 | 0.945 | 0.947 | | 496024 |

Overall, the CNN model achieved a high accuracy of 94.6% on the full dataset. The model performed well across the classes, as seen by the weighted averages of precision, recall, and F1-score (all values are available at Table 7).

The CNN performed remarkably well for the 'Unchanged' class, which has the highest number of training points in the dataset, achieving an accuracy of 0.968 and a recall of 0.971. These values indicate that the model was very accurate in predicting this class.

The model performed very well in both "New Building" and "Demolition" classes, achieving an F1-scores of 0.897 and 0.875. These results suggest that the CNN was able to learn relevant features from the dataset for these classes, leading to a high rate of correct predictions.

The 'New Vegetation' class had the lowest precision of 0.372 but a high recall of 0.925. The low precision shows that the model had many false positives for this class, while the high recall shows that it was able to correctly identify a large proportion of the actual 'New Vegetation' instances.

The 'Vegetation Growth' class didn't perform that well, with an F1-score of only 0.672. Also with an F1-score of 0.774, the 'Vegetation Loss' class showed comparable results.

The 'Mobile Objects' class showed a perfect performance, however it should be noted that this class had the least support in the dataset, which might have contributed to this outcome.

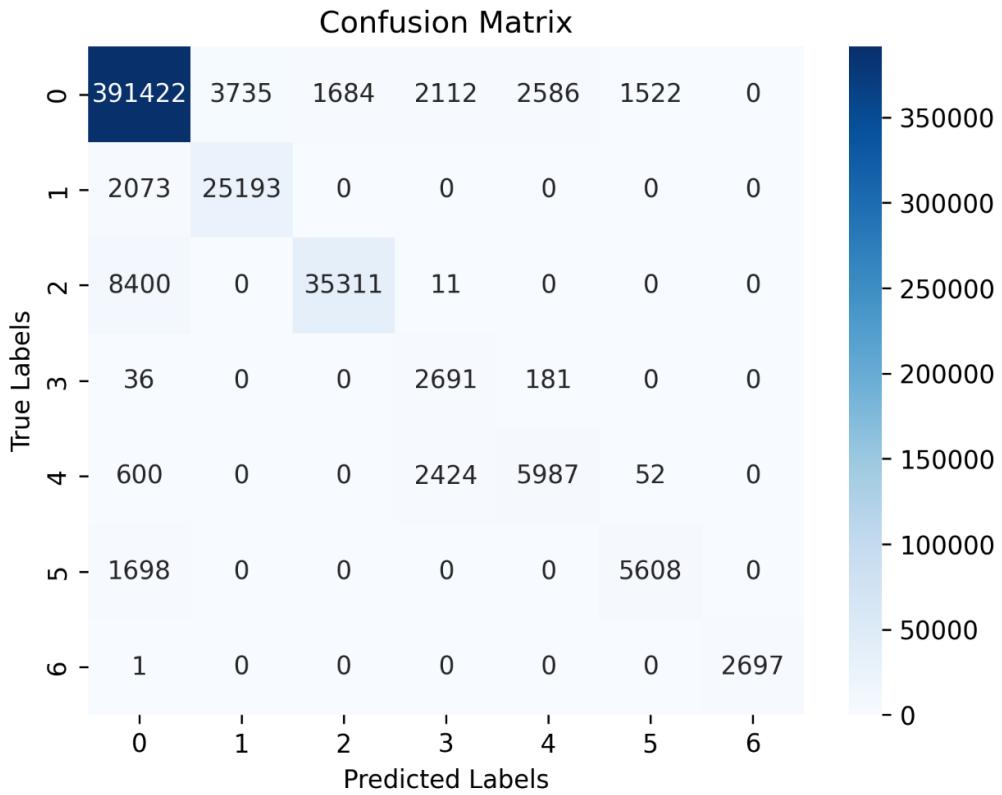


Figure 36 - Confusion matrix of CNN model trained on the full dataset

b. Half dataset trained CNN

All the accuracies and confusion matrixes for both of the half datasets are available on the “Appendix” section.

The CNN trained on the first half of the dataset performed really well, with an overall accuracy of 94.5%. The 'Unchanged' class had an F1-score of 0.970, with a high precision and recall. Similar to the full dataset results, the model performed well on the 'New Building' and 'Demolition' classes, with F1-scores of 0.895 and 0.876. The 'New Vegetation' class still had a low precision leading to an F1-score of 0.529. The results of 'Vegetation Growth' and 'Vegetation Loss' classes were slightly lower than in the full dataset, while the 'Mobile Objects' class maintained perfect performance.

The performance of the CNN on the second half of the dataset was similar to the first half, with an overall accuracy of 94.6%. The 'Unchanged' class performance slightly improved, while the 'New Building' and 'Demolition' classes maintained the same F-score as the full dataset. With an F1-score of 0.529, the 'New Vegetation' class continued to display a significant difference between the accuracy and the recall. While "Vegetation Loss" performed similarly to the first half dataset, with an F1-score of 0.776, the "Vegetation Growth" class did better in this dataset with an enhanced F1-score of 0.606. The 'Mobile Objects' class once again showed perfect performance.

CNN Analysis

The CNN model showed consistency in its performance across the full, first half, and second half datasets. The overall accuracy remained between 94.5% and 94.6%. Our model achieved high F1-scores on classes with high support, such as "Unchanged," "New Building," and "Demolition." These results show that the model has learned the underlying patterns of the data.

The improvement in precision, when compared to the other two models, for the 'Vegetation Growth' means that the model has learned some distinctive spatial features that the other models were unable to. This is an interesting avenue for further investigation, as this was the class with the lowest accuracy value. Understanding these features could provide insights into the nature of such classes and improve their detection.

However, the model struggled with the 'New Vegetation' class, but performed much better than other models. The model showed lower accuracy with classes that have fewer instances and it is possible that it has bias towards classes with more instances. This is a common issue in machine and deep learning. CNN model's complexity, while enabling it to model complex patterns in the data, also makes it more susceptible to overfitting.

In conclusion the CNN model was able to learn from the dataset and make accurate predictions. The differences in performance between classes, particularly for the "New Vegetation" and "Vegetation Growth" classes, indicate that there may be room for further optimization.

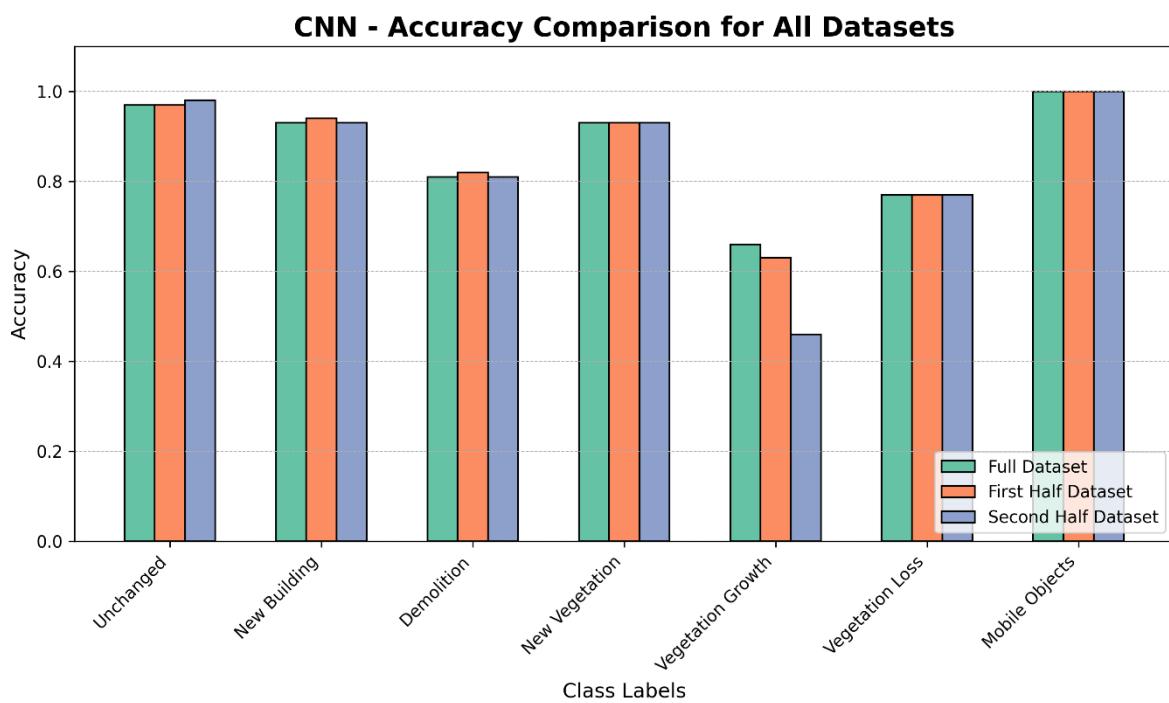


Figure 37 - Per-class accuracy of all datasets for the CNN model, Urb3DCD

5.2. Results for AHN Dataset

In this section, we will talk about the results obtained from our three change detection methods: Random Forest (RF), Fully Connected Neural Network (FCNN), and Convolutional Neural Network (CNN), to the AHN dataset. The objective of these experiments is to evaluate the performance of each model.

However, after a visual investigation of the results on the CloudCompare, we noticed that the pattern of the errors for all three of our models was similar to the process of creating the training, validation and testing data. Multiple small clusters were misclassified as random changes. So, just as we did during the data preparation process, to improve the performance of the classifier, a clustering algorithm was applied post-classification. This part of the code was designed to handle the issue of misclassified random points in the dataset. Specifically, if a cluster of a certain label was very small, it was reassigned to the closest label spatially. We will talk more about this on the “Comparative Analysis and Visualization of the Results” section. The following subsections will provide a numerical overview of the results for each method.

Table 8 - AHN classes distribution

| Class Name | Class Number | Points in the dataset | Counts of elements in percentage |
|--------------|--------------|-----------------------|----------------------------------|
| Unchanged | 0 | 641373 | 76.34% |
| New building | 1 | 113756 | 14% |
| Demolition | 2 | 85027 | 10% |

5.2.1. Random Forest Classifier

Table 9 -Accuracies of RF model trained on the AHN dataset

| Class | precision | recall | f1-score | IoU | support |
|--------------|-----------|--------|----------|------|---------|
| Unchanged | 0.96 | 0.97 | 0.97 | 0.94 | 1008648 |
| New building | 0.8 | 0.71 | 0.75 | 0.60 | 62914 |
| Demolition | 0.64 | 0.63 | 0.63 | 0.46 | 51681 |
| Accuracy | | | 0.94 | | 1123243 |
| macro avg | 0.8 | 0.77 | 0.78 | | 1123243 |
| weighted avg | 0.94 | 0.94 | 0.94 | | 1123243 |

The RF model that was applied to the AHN dataset, consists of three target classes: Unchanged, New Building, and Demolition. The initial accuracy of the RF classifier was 92.38%. After we applied our cluster cleaning code, it effectively cleaned up the output of the RF classifier, enhancing the accuracy to 94.00%.

The results shown in the Table 9 indicate that the RF classifier, combined with the clustering algorithm, performed best on the Unchanged class, with high a precision of 0.96, recall 0.97, and F1-score 0.97. The performance on the New Building class was also relatively good with a precision of 0.8. However, the performance on the Demolition class was low, indicating that the classifier had some difficulty accurately identifying this class.

As this algorithm cannot get more complex features out of the features we provide, it is bound to remain relatively simple compared to more sophisticated deep learning models. The RF classifier, when combined with a post-classification clustering algorithm, demonstrated strong performance on the AHN

dataset, particularly in identifying unchanged areas, but there is room for improvement in the classification of new buildings and demolished areas. The misclassification issue between unchanged trees and changed buildings highlights the need for more specialized approaches to handle such challenges in change detection tasks.

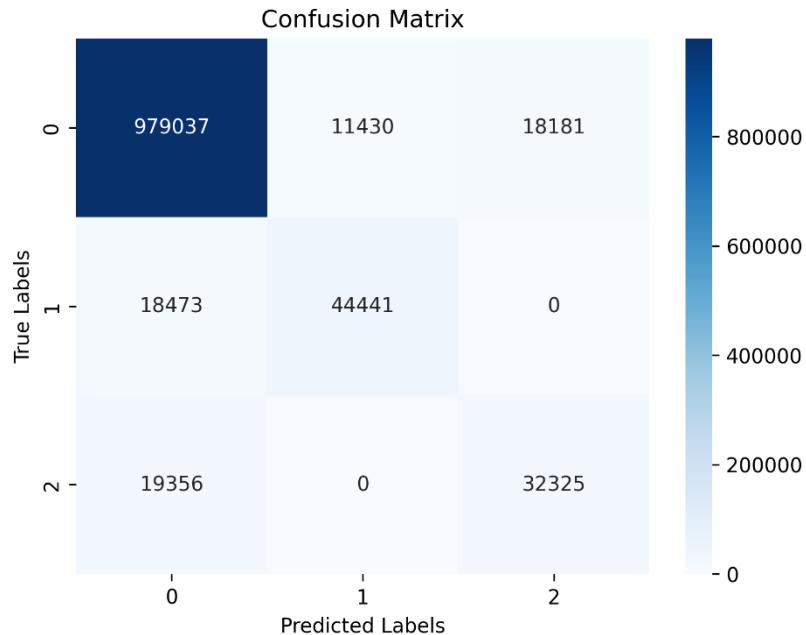


Figure 39 - Confusion matrix of RF model trained on the AHN dataset

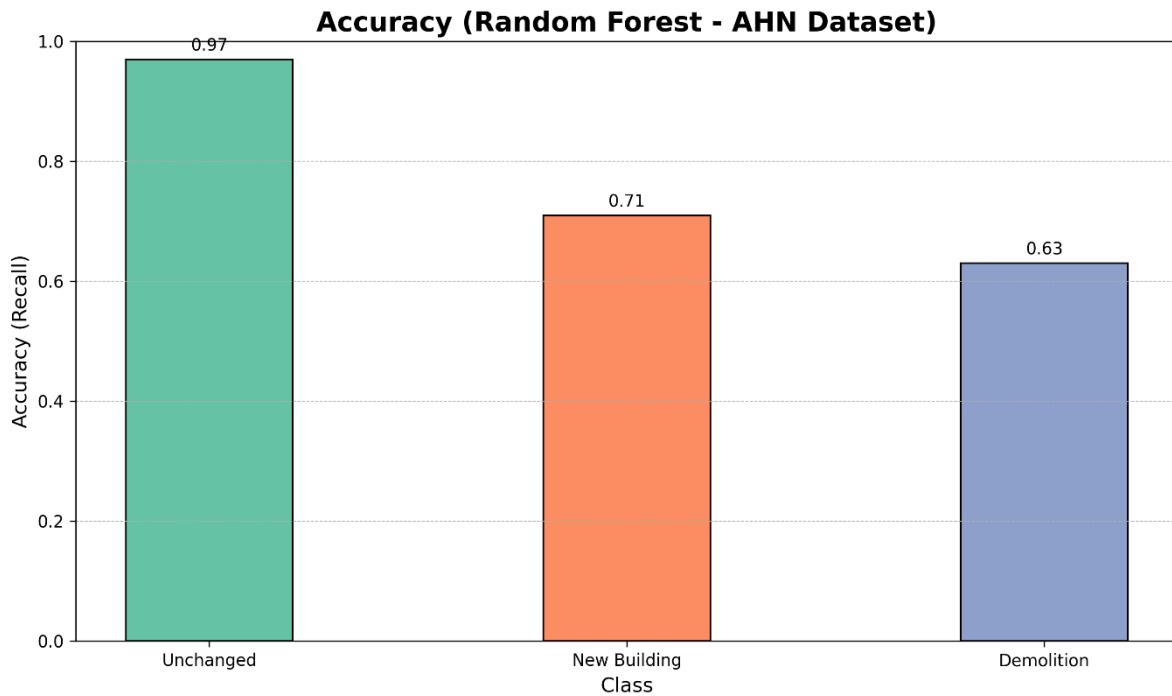


Figure 38 - Per-class accuracy for the RF model, AHN dataset

5.2.2. Fully Connected Neural Network

The FCNN was used to classify three classes in the AHN dataset: Unchanged, New Building, and Demolition. After applying a clustering code post-classification, the accuracy of the FCNN increased from 92.65% to 93.66%. The precision, recall, and F1-score for each class are shown in the Table 10.

Table 10 - Accuracies of FCNN model trained on the AHN dataset

| Class | precision | recall | f1-score | IoU | support |
|--------------|-----------|--------|----------|------|---------|
| Unchanged | 0.96 | 0.97 | 0.96 | 0.93 | 1008648 |
| New building | 0.82 | 0.64 | 0.72 | 0.56 | 62914 |
| Demolition | 0.6 | 0.68 | 0.64 | 0.47 | 51681 |
| accuracy | | | | 0.94 | 1123243 |
| macro avg | 0.79 | 0.76 | 0.77 | | |
| weighted avg | 0.94 | 0.94 | 0.94 | | |

For the 'New Building' class, the model achieved a precision of 0.82 and a recall of 0.64, leading to an F1-score of 0.72. While the precision is relatively high, indicating that a large proportion of instances classified as 'New Building' were correct, the lower recall suggests that the model missed a significant number of 'New Building' instances. The 'Demolition' class had a precision of 0.60 and a recall of 0.68, resulting in an F1-score of 0.64. This indicates that the model had some difficulty in correctly classifying 'Demolition' instances, with a significant number of instances either being missed (as indicated by the recall) or being misclassified (as indicated by the precision).

The macro average scores were 0.79 for precision, 0.76 for recall, and 0.77 for the F1-score. These results show that the model performed similarly across all classes. The 'Unchanged' class had an impact on the overall performance, as seen by the weighted average scores, which give more weight to the classes with more instances, getting a value of 0.94 for all scores. In summary, the FCNN model performed well on the AHN dataset, particularly for the 'Unchanged' class. However, the results also highlight areas for potential improvement, particularly for the 'New Building' and 'Demolition' classes.

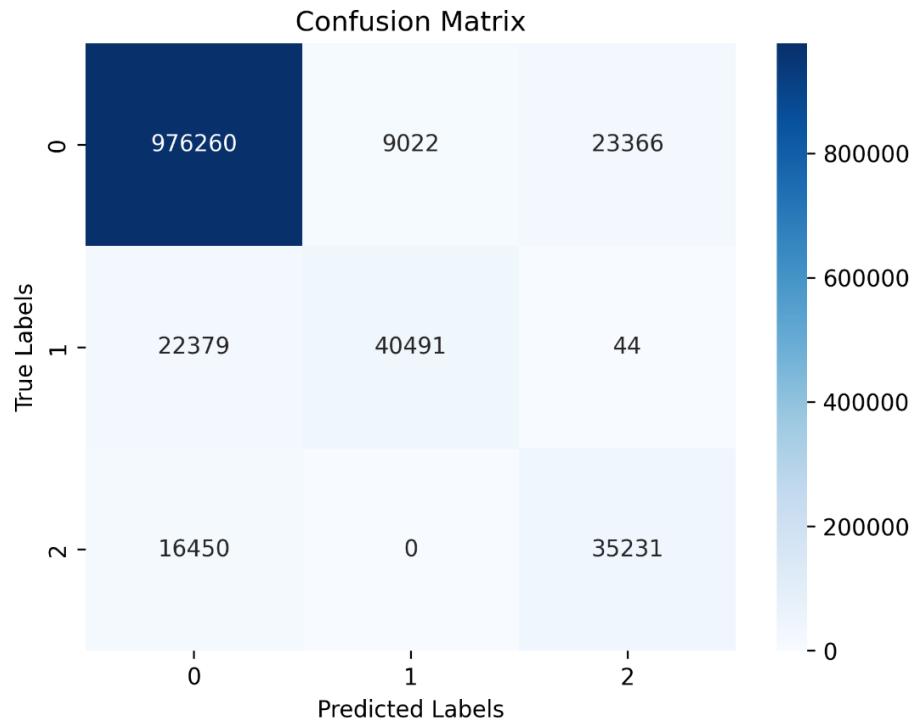


Figure 40 - Confusion matrix of FCNN model trained on the AHN dataset

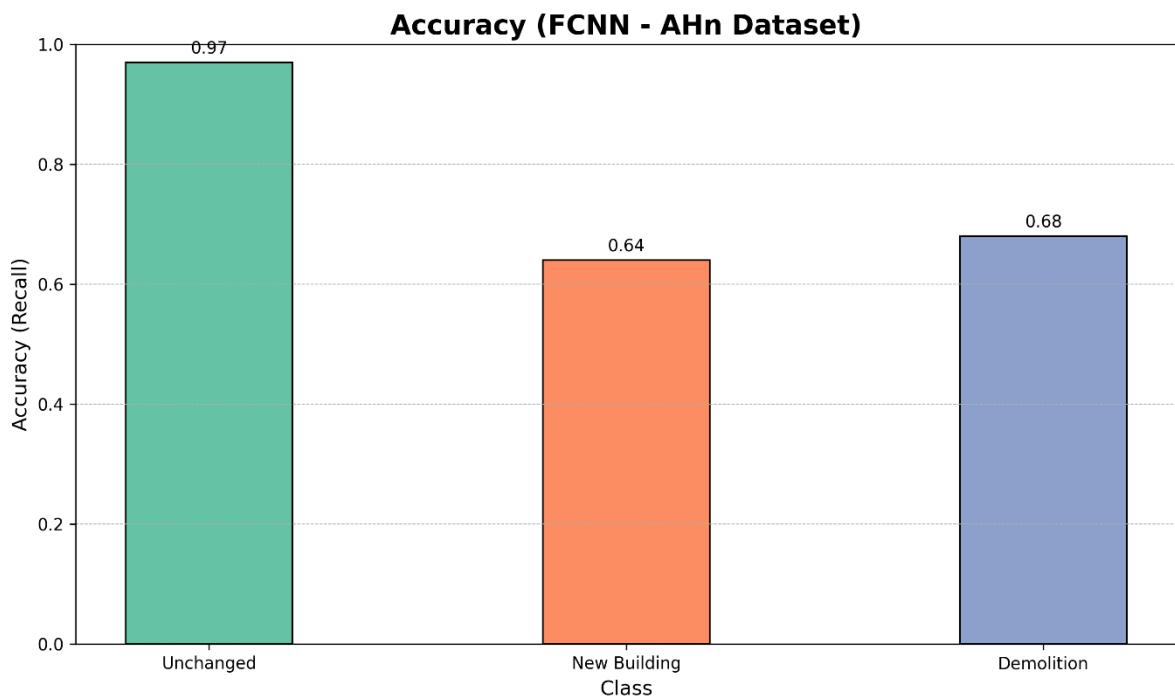


Figure 41 - Per-class accuracy for the FCNN model, AHN dataset

5.2.3. Convolutional Neural Network

After applying a clustering algorithm post-classification, similar to the approach used with the RF and FCNN, the accuracy of the CNN increased from 92.25% to 93.79%. The precision, recall, and F1-score for each class are shown in the Table 11.

The most represented class in the dataset, in our case the 'Unchanged' class, had a precision of 0.96 and a recall of 0.97, and F1-score of 0.97. The CNN model was successful in correctly identifying 'Unchanged' points and also avoided the misclassification of other classes as 'Unchanged'.

For the 'New Building' class, the model achieved a precision of 0.79 and a recall of 0.62, resulting in an F1-score of 0.69. The relatively high precision indicates a large amount of points classified as 'New Building' were indeed correct, while the lower recall indicates that the model failed to identify a significant number of 'New Building' points.

The 'Demolition' class had a precision of 0.63, a recall of 0.77 and an F1-score of 0.69. The model had some issues in accurately classifying 'Demolition' instances, with a significant number of points either being overlooked or being misclassified.

The macro average scores, which consider each class equally, were 0.79 for precision, 0.79 for recall, and 0.78 for the F1-score. These scores suggest a relatively balanced performance of the model across the classes. The weighted average scores, which give more weight to the classes with more instances, were all 0.94. This shows the strong influence of the 'Unchanged' class on the overall performance.

Table 11 - Accuracies of CNN model for AHN dataset

| Class name | Precision | Recall | F1-score | Iou | Support |
|--------------|-----------|--------|----------|------|---------|
| Unchanged | 0.96 | 0.97 | 0.97 | 0.93 | 1008648 |
| New building | 0.79 | 0.62 | 0.69 | 0.53 | 62914 |
| Demolition | 0.63 | 0.77 | 0.69 | 0.53 | 51681 |
| accuracy | | | 0.94 | | |
| macro avg | 0.79 | 0.79 | 0.78 | | 1123243 |
| weighted avg | 0.94 | 0.94 | 0.94 | | 1123243 |

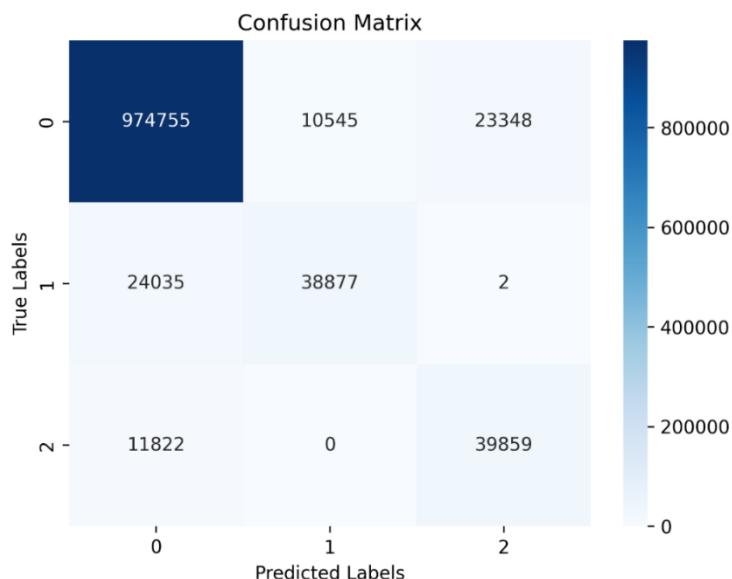


Figure 42 - Confusion matrix of CNN model trained on the AHN dataset

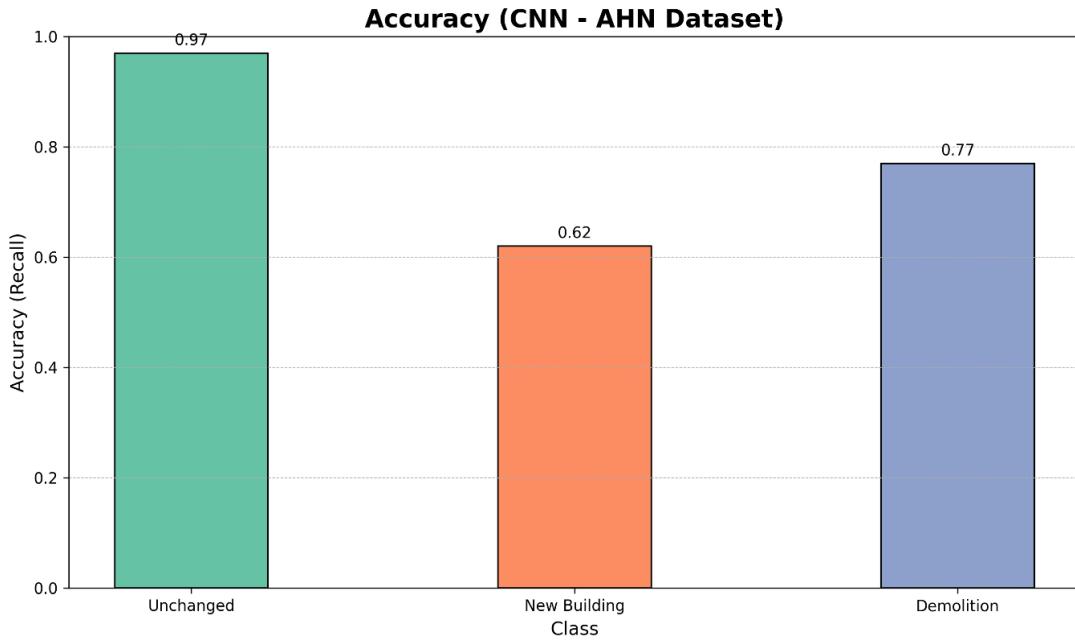


Figure 43 - Per-class accuracy for the CNN model, AHN dataset

6. DISCUSSION

6.1. Comparative Analysis and Visualization of our Results

6.1.1. URB3DCD Dataset

The Urb3DCD dataset, a simulated 3D point cloud dataset, served as the testing ground for three different change detection models: Random Forest (RF), Fully Connected Neural Network (FCNN), and Convolutional Neural Network (CNN). Comprising 1,638,678 points (3.6km^2 per epoch), the dataset includes a variety of class labels that represent diverse types of changes within an urban environment. A direct comparison of each class accuracy is illustrated in the Figure 44.

The RF model, as a traditional machine learning algorithm, relies heavily on the basic and hand-crafted features, as it does not build higher-level features from the input. The Random Forest algorithm in our scenario only utilizes RGB information and the change in height. It is not uncommon to classify changes on 2D images based only on their RGB values and they have shown very good results in the past (Nemoto et al., 2017). Our theory that the same process can be done in point clouds, looking at the promising results, was proved right. The model was able to learn patterns in RGB and z changes and correctly identify almost all of the changes in the target elements. It exhibited exceptional performance on classes with high support as these classes were well-represented in the training data, which is reflected in the validation results. However, it encountered difficulties with the 'New Vegetation' class. This could be attributed to the fact that the 'New Vegetation' class and "Vegetation growth" possesses features that are not easily distinguishable by the RF model, as these two classes have majority of feature in common. Their RGB values will remain relatively the same and only a small difference in z will be prevalent, which is one of the reasons the algorithm had some difficulties with these classes. It is also possible that the training dataset is not representative enough of the validation dataset. Therefore, if these features are not sufficiently represented, errors, as shown in the Figure 46 (b), will occur. Many trees were not correctly classified, so even though RF works in most of the urban elements, there is still room for improvement.

The Fully Connected Neural Network (FCNN) model also demonstrated strong performance on the Urb3DCD dataset. FCNNs, being a type of neural network, have the ability to learn complex patterns in the data. Different from RF, this model was able to craft more complex features out of the input initial features we provided. Thus as seen in the Figure 46 (c), even though this algorithm is still misclassifying some random points around the buildings or in the ground as changes, it showed a much better understanding of different classes and classified the underrepresented classes better. The model did not confuse the similar classes and showed very good predicting abilities for different classes. The ability to craft more complex features made it possible to find differences and correctly classify even very similar classes. There is a considerable decrease in random errors misclassified as change when we compare FCNN with RF. Even though we saw an improvement when we jumped from RF to FCNN, they both showed some errors when we visualized the output. The result numbers are very satisfying, however, the algorithms' lack of capacity to account for spatial relationships presents a significant limitation, particularly in dealing with spatially-related issues such as those encountered in our study. Despite the spatial aspect of this simulated dataset not introducing significant errors, the models' disregard for their surrounding context is still a potential source of errors.

Our FCNN and RF models are trained in a point-wise manner, focusing only on how individual points shift from one epoch to another without considering any additional factors. This approach, unfortunately, is a significant source of errors, especially in denser datasets where mistakes can occur more often. If the model fails to recognize that our data points aren't isolated entities classified in a sequential manner, the denser the dataset, the higher the likelihood of errors and misclassifications. For this reason, we shift our attention to the CNN approach in the following paragraph, a model designed to also consider the surrounding points of our target point to make a more informed prediction.

The Convolutional Neural Network (CNN) model showed the highest accuracy overall and a consistency in its performance. CNNs, unlike FCNNs and RF, can capture the spatial relationships between the data points, which could explain their better prediction. As shown in the Figure 46 (d), the trees are mostly classified correctly, much better than the other models. Also if we notice in the close up figure, almost all the small misclassified errors in the ground or around buildings that the other two models have, disappeared in the CNN prediction. As CNN uses spatial information, it is able to understand the properties of the surrounding area, thus allowing it to make a better prediction and avoiding a lot of misclassification of random points. The model has a better understanding of the element it's classifying as it is considering also how the neighbourhood of each point changed between epochs. It is not only calculating higher-level features of the predicted point, but it's also combining the features of the neighbouring points. Thus even though the CNN is a more complex and computationally heavy model, the trade-off is definitely worth it.

In conclusion, all three models demonstrated strong performance on the URB3DCD dataset, with the CNN model showing the highest overall accuracy. However, all models showed room for improvement, suggesting that the feature set used for training might need to be expanded or refined to better representation. Furthermore, the models could potentially benefit from additional training data for the underrepresented classes.

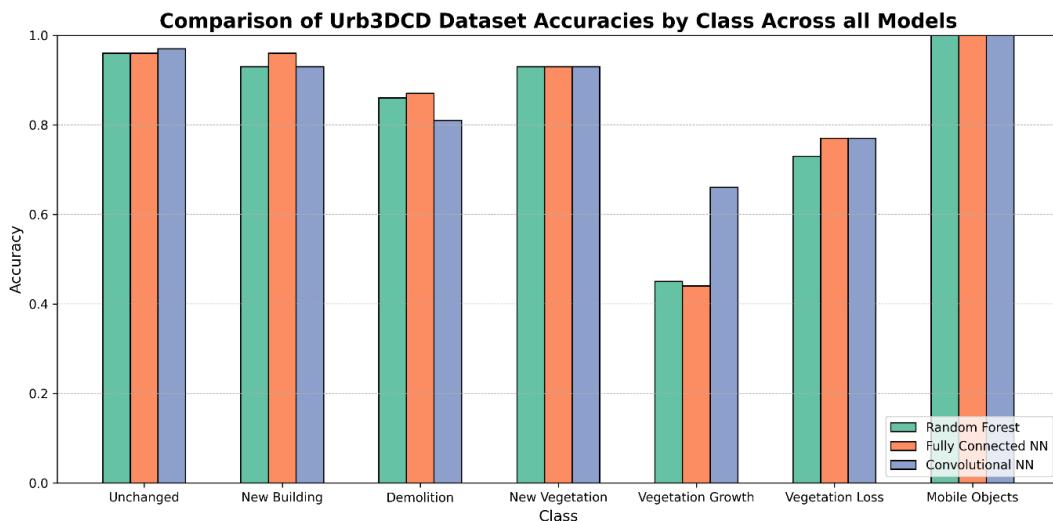
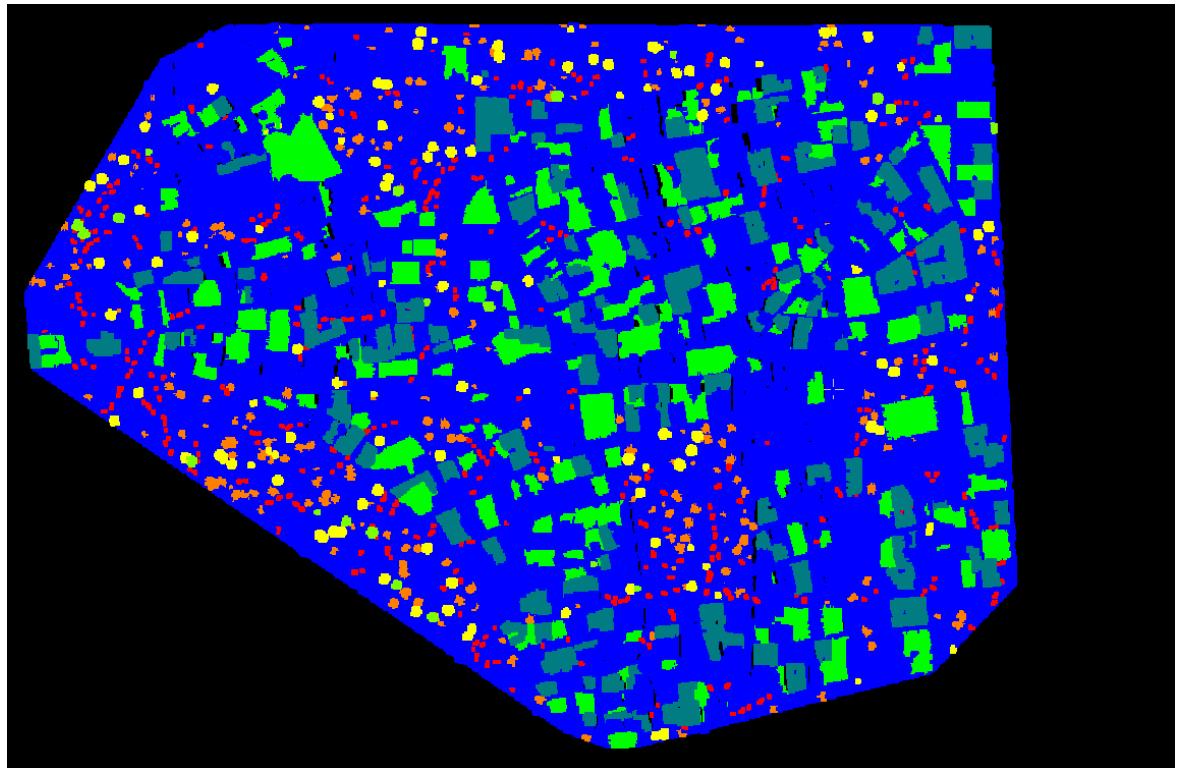
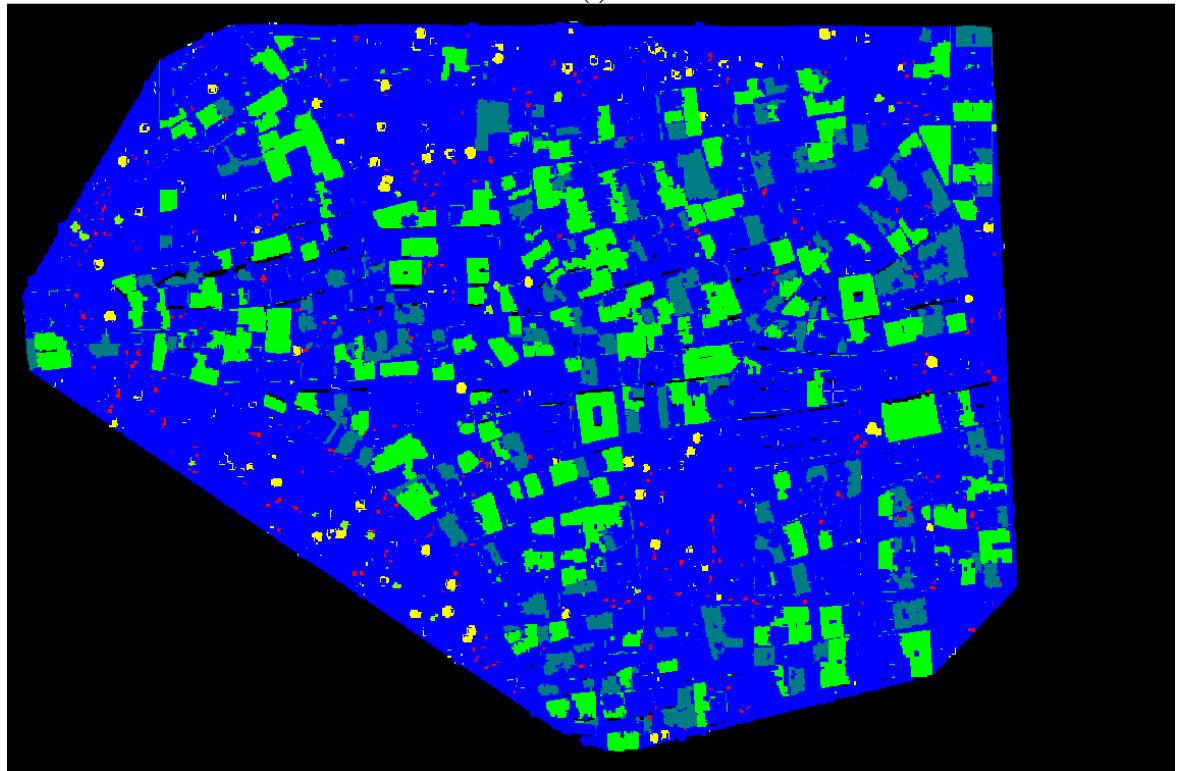


Figure 44 - Per-class accuracy of all the models, Urb3DCD



(a)



(b)

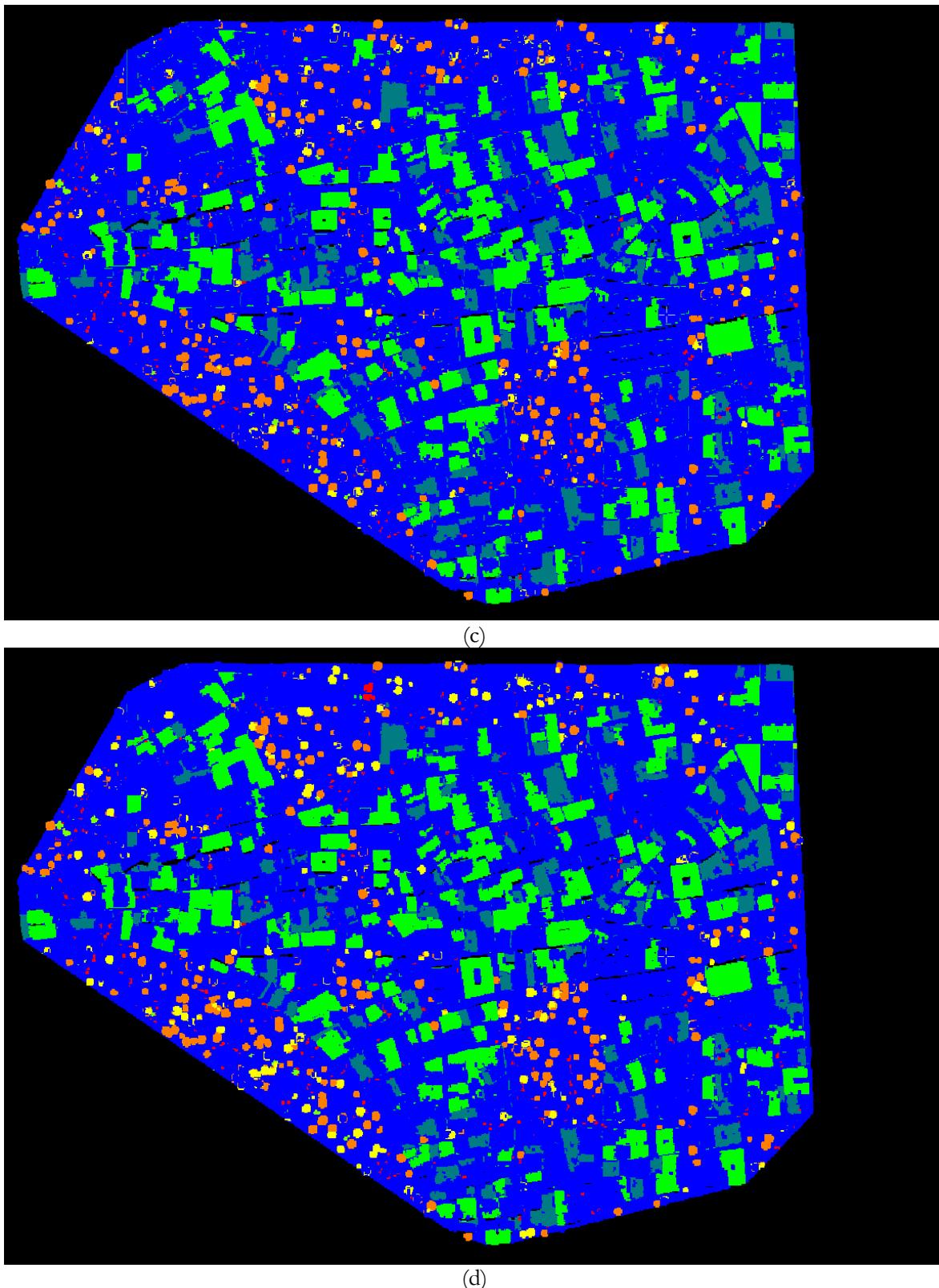


Figure 45 - A predicted tile from the Urb3DCD dataset. (a) Ground truth, (b) Random forest, (c) Fully Connected NN, and (d) Convolution NN. Blue colour represents non-change, dark green colour is new building, green is demolition, light green is new vegetation, , yellow is vegetation growth , orange is missing vegetation, and red is mobile object.

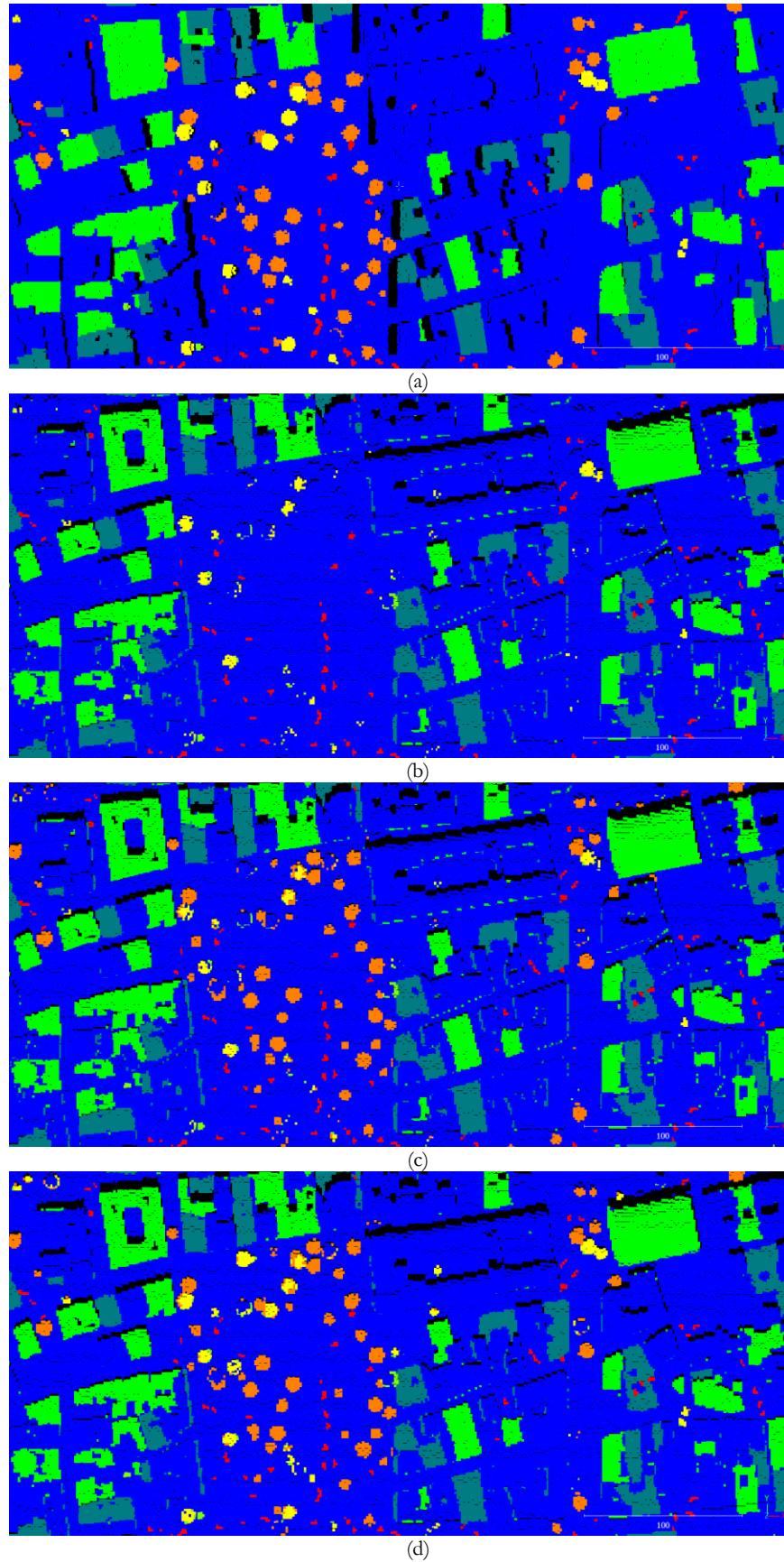


Figure 46 - Close up view of Urd3DCD dataset results. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN. Blue colour represents non-change, dark green colour is new building, green is demolition, light green is new vegetation, , yellow is vegetation growth , orange is missing vegetation, and red is mobile object.

6.1.2. AHN dataset

The Actueel Hoogtebestand Nederland (AHN) dataset, a real-world 3D point cloud dataset, was used to test three different change detection methods: Random Forest (RF), Fully Connected Neural Network (FCNN), and Convolutional Neural Network (CNN). The dataset consists of three changed classes: Unchanged, New Building, and Demolition.

As mentioned before, after we predict the changes with the algorithms, a cluster cleaning code is applied. Figure 47 (a) shows the original output from the model, and its noticeable that a lot of non-changed points are misclassified as changes. Figure 47 (b) is the outcome of our cluster cleaning, which changes the label of small clusters with the biggest nearest cluster. The idea behind this is that the changes in buildings cannot be a small group of points, especially in such a densely populated point cloud. As it can be seen in the Figure 47, it was really successful of making the predicted tile better and removing a lot of errors. The overall accuracy for all three models was increased by a good amount. As mentioned in the methodology section, RGB values were not reliable so we ended up using only the intensity, return number, z and point density in 2D as our input features for the models. Even though these values were correct and really showed some good results in CNN if we look at the statistical values, they were not enough to ensure a clean classification just like in URB dataset where CNN didn't misclassified random points.

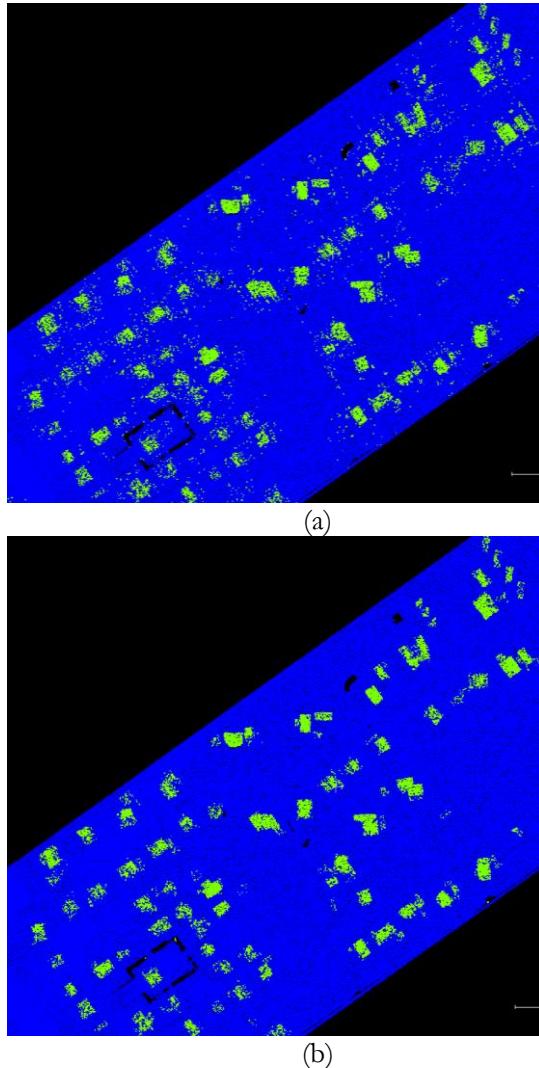


Figure 47 – A close look into a predicted test tile of the AHN dataset. (a) Original CNN output and (b) Cleaned CNN output. Blue colour represents non-change, yellow/light green represents new buildings.

The Random Forest (RF) model demonstrated the highest accuracy percentage wise, however if we have a closer look into the output, we will notice that this is because of classes imbalance. The unchanged class has the highest amount of points, thus it is affecting the overall results more than the other two classes.

RF faced the issue of misclassifying unchanged trees, unchanged buildings and ground as changed buildings. This is shown in Figure 50 (b), where it is clear that without crafting extra features and taking neighbours into consideration, it is really easy to misclassify points if we consider each of them as a standalone. Even though we added the number of returns and crafted the feature of the density of point clouds in a 2D radius, which in theory should help with this problem, it was still prevalent. This could be attributed to the similarities in the feature space between these classes, which might have caused confusion for the classifier. This issue points to the limitations of the RF classifier in distinguishing changes between urban elements. As mentioned before, the RF and FCNN models classify the points in a one to one manner, thus taking into account only the features they “see” on individual points. The ‘Unchanged’ and ‘New Building’ classes might have features that are not easily distinguishable by the RF model, so its source of distinction between these classes is very limited. Also we need to consider that we are only using z, intensity, return number and point density in 2d as features. The main feature to distinguish all the classes in previous classification tasks was RGB, which in our case is missing. So in our case, this role is most likely done by intensity which in different parts of the Netherlands will have different values as the roofs of the buildings and the type of the vegetation changes, leading to a different value. This can cause errors when trained and tested in two locations far away from each other.

The Fully Connected Neural Network (FCNN) model also demonstrated strong performance on the AHN dataset. Even though it still misclassified trees just like RF in many instances, it didn’t misclassify buildings, as the higher dimension features it creates did come handy in this scenario. This means that crafting more complex features helps a lot in distinction between very similar objects. The problem of the trees in this model is most likely the fault of the one-to-one classification. The one to one matching is done on a 2D plane, thus there are a lot of instances where tree points of one epoch are matched with ground points from the other epoch, causing a huge difference in the z feature, which leads into many points classified as change. This issue however is not seen in CNN model output as seen in Figure 50 (d).

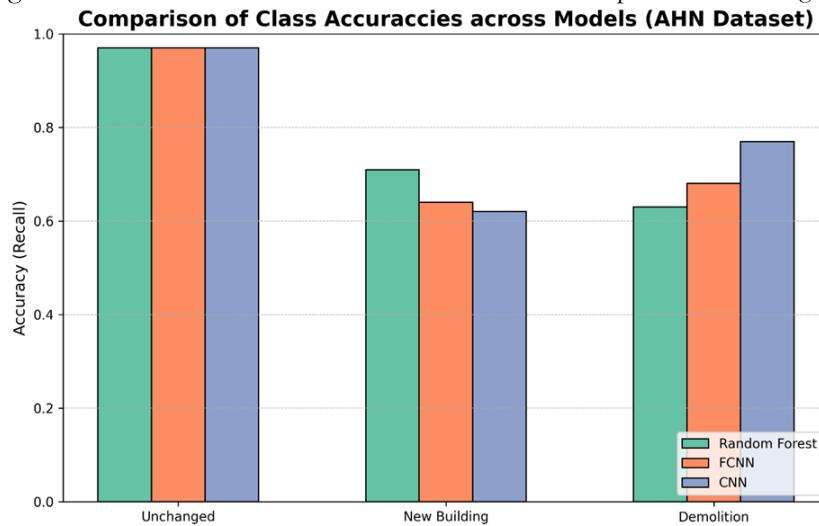


Figure 48 - Per-class accuracy of all the models, AHN dataset

Finding nearest point in 3D requires much more computational power, and when we have to deal with huge and very dense data just like AHN, it will take too much time and additional complexity. Also since our main focus was buildings, which change only on planar level, we decided to go with it. Most likely 3D can potentially work a bit better for the other non-spatial models, but ultimately we wanted to prove the efficiency of CNN and keep the model as fast as possible, we choose to go with the 2D matching. While the matching process indeed occurs in 2D, based on the closest X and Y values, it doesn’t impact the other calculations. The X and Y values are just used as an indication of closest points, without any interference on how we proceed further. In the context of the CNN model, the identification of neighbouring points still considers the full 3D space, taking into account all features of these points.

The Convolutional Neural Network (CNN) model showed an overall accuracy of 93.79% on the AHN dataset. CNNs can capture the spatial relationships between the data points. Unlike the RF and FCNN models, the CNN model did not misclassify trees or buildings. Even though as we mentioned there are a lot of instances where tree points were matched with ground points, the CNN showed that it can understand that there is no change in that area by looking at the neighbouring points. The ability to consider the area

around the points and craft complex features made it possible to have a minimal amount of points misclassified. It still missed some points of buildings inside the building cluster but there was no visible error in the predicted classes. This suggests that the CNN was more successful in distinguishing between natural and built environments compared to the other classifiers.

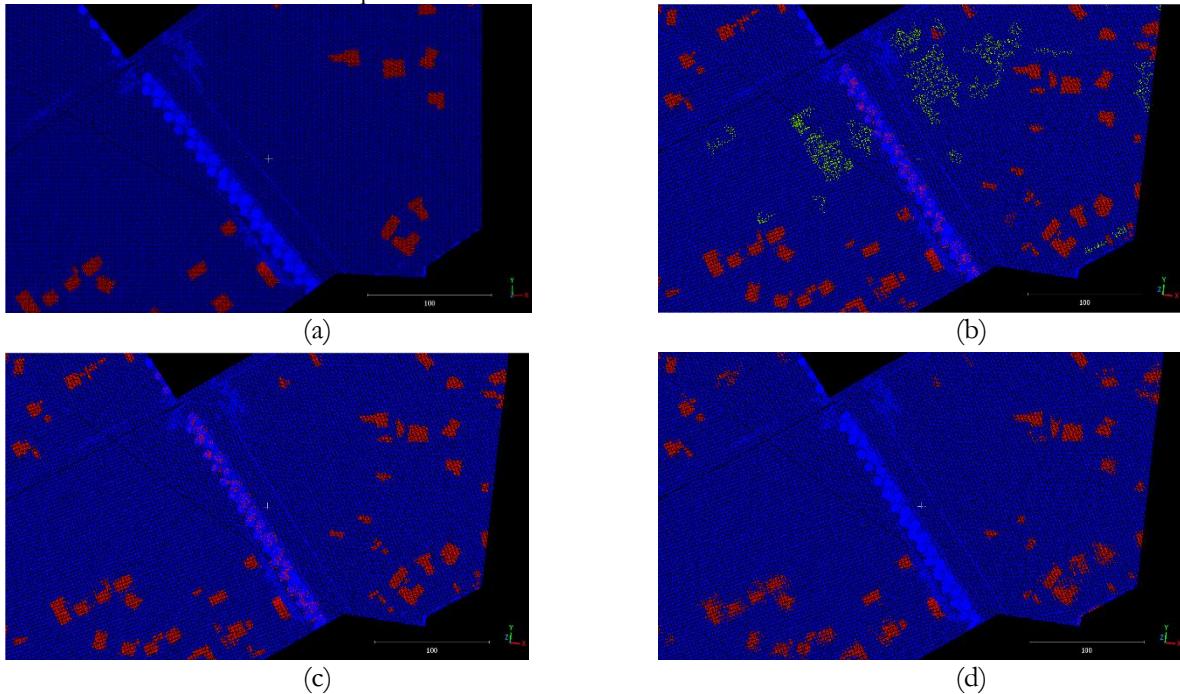


Figure 49 - Prediction on an AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN. Blue colour represents non-change, yellow/ light green represents new buildings and red represents demolition.

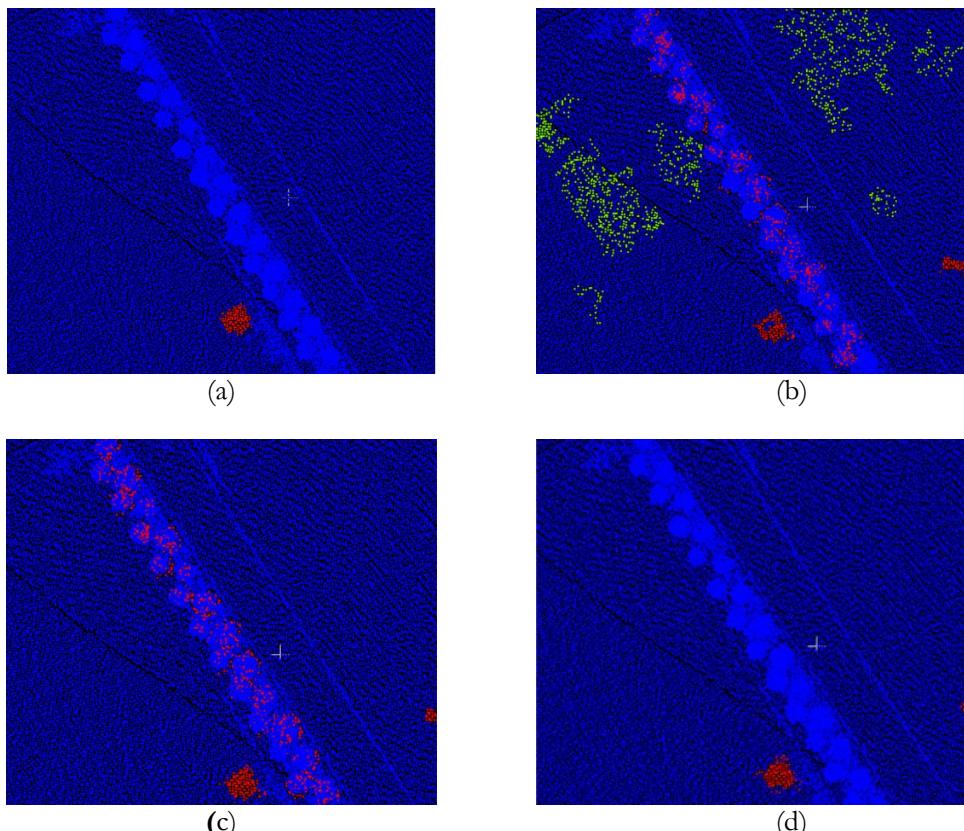


Figure 50 - A closer look into the prediction of the AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN. Blue colour represents non-change, yellow/ light green represents new buildings and red represents demolition.

However we do have some interesting scenarios too. In the Figure 52 we can see that the Random Forest model made a much cleaner outline of the buildings than the deep learning models. This is most likely because the model was trained in a similar tile where the type of building roofs are very similar to the validation data, thus showed the superiority of RF when it comes to predicting well represented areas, while CNN tried to overcomplicate the problem and using neighbouring points, and made mistakes on the points next to the buildings.

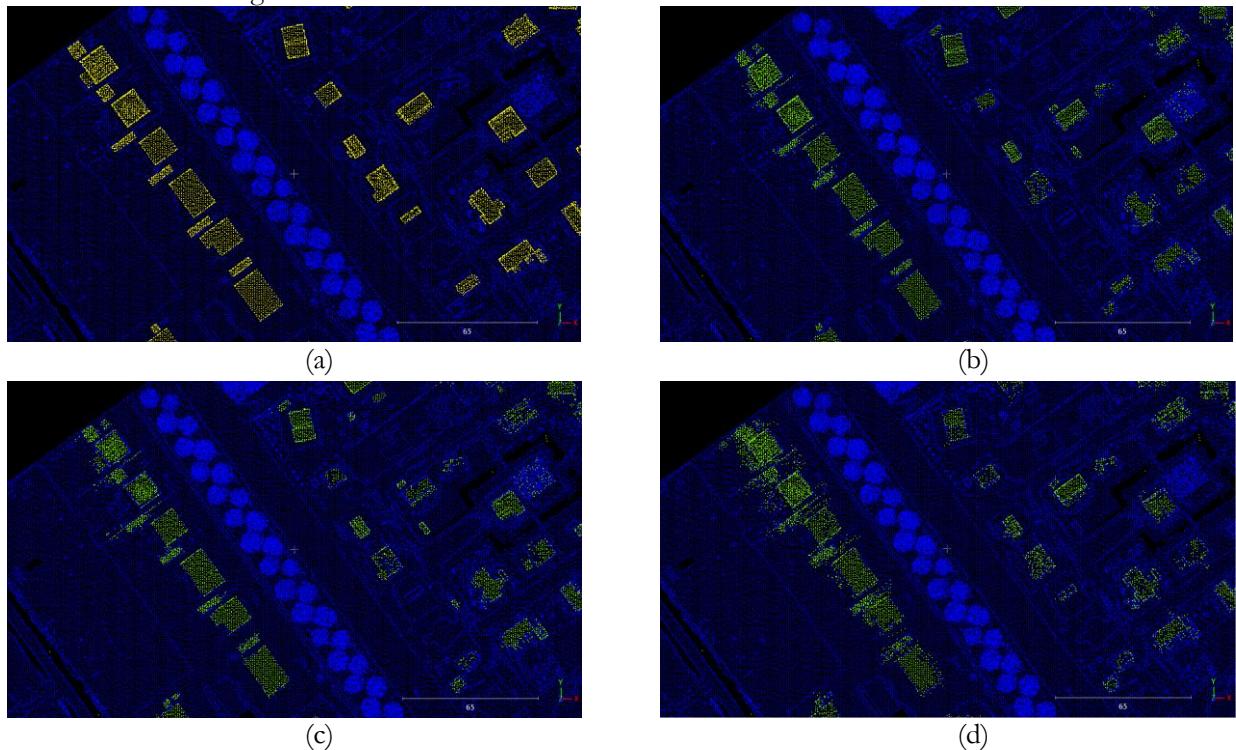


Figure 51 - Prediction on another AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN. Blue colour represents non-change, yellow/ light green represents new buildings.

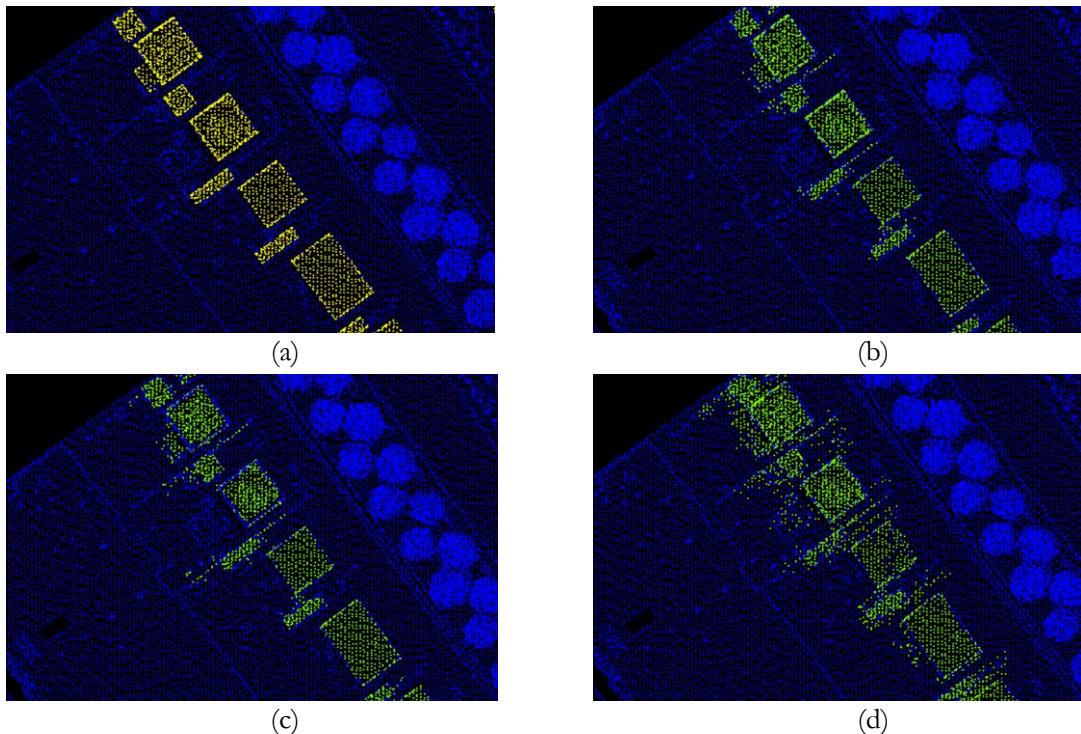


Figure 52 - A closer look at the prediction of the tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN. Blue colour represents non-change, yellow/ light green represents new buildings.

Another interesting scenario is when we examine an area that significantly differs from the training data. As shown in Figure 53, the RF fails to identify the correct changes, while the deep learning models, by crafting complex features, understand the problem better and make a more educated prediction. However, we can also notice that the CNN, through its utilization of neighbouring points, adds unnecessary complexity around the building areas. While this spatial awareness has addressed many issues related to other classes, it can also introduce unnecessary complications that result in errors.

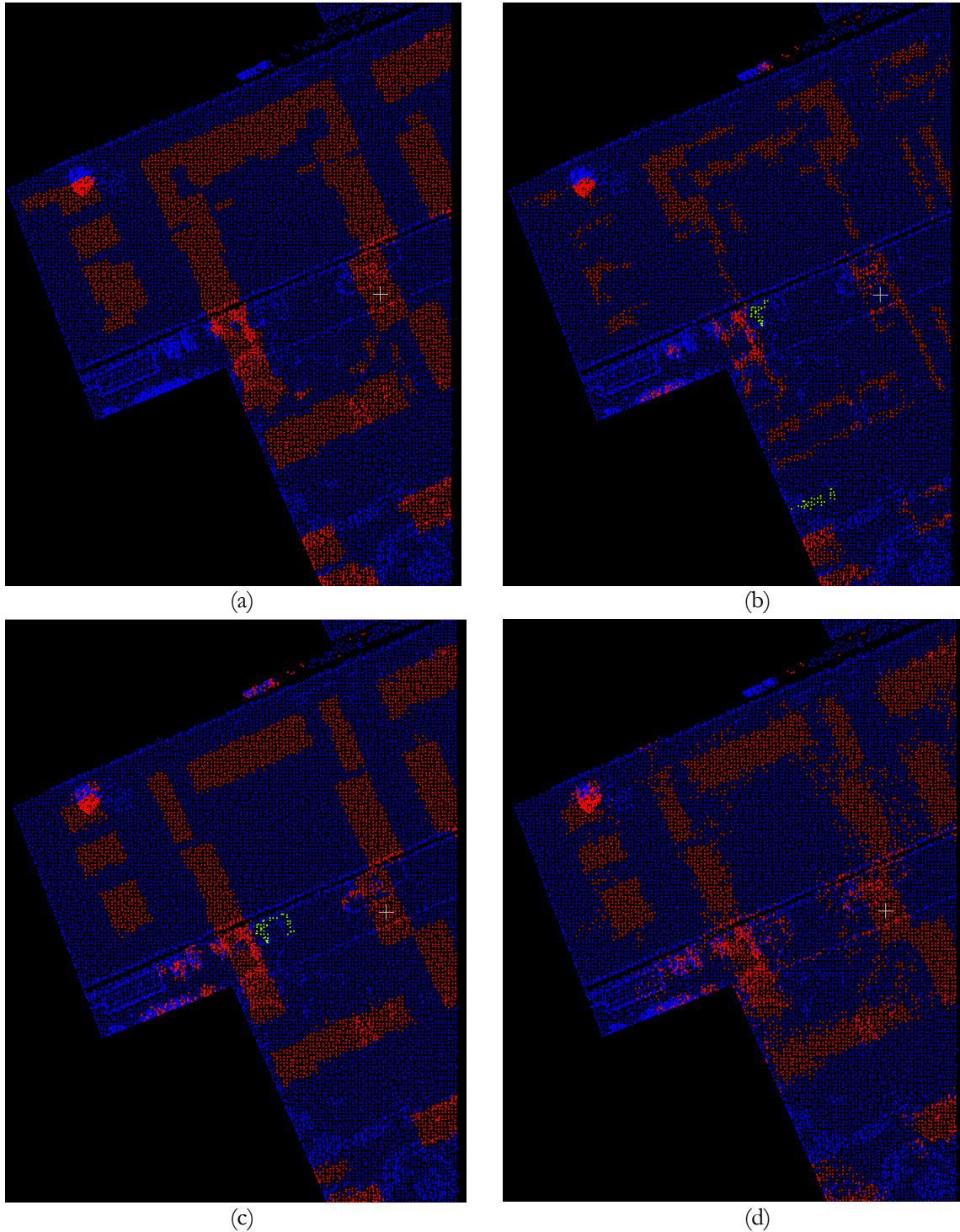


Figure 53 - Predicted AHN tile. (a) Ground truth, (b) RF, (c) FCNN, and (d) CNN. Blue colour represents non-change and Red colour represents demolition.

In conclusion Random forest is a fast and reliable algorithm if we have a well-represented training dataset. However, its shortcomings become evident in its inability to comprehend spatial context, poor performance in regions that are very different from the training data, and potential confusion between similar classes due to the absence of higher-level features. On the other hand, deep learning models demonstrate greater robustness, although their complex features can sometimes overcomplicate certain scenarios. Out of all models examined, the CNN model stands out as the most reliable and consistent for dealing with spatial problems. All three models demonstrated relatively good performance on the AHN dataset. The models could potentially benefit from additional training data for the problematic classes. The CNN's success in accurately classifying buildings and not misclassifying trees highlights the potential of this approach for complex change detection tasks.

6.1.3. Comparison between URB3DCD and AHN Datasets

The Urb3DCD and AHN datasets, while both being 3D point cloud datasets, present different challenges and characteristics that affect the performance of the change detection methods. The Urb3DCD dataset is a simulated dataset with a variety of class labels representing seven types of changes in an urban environment. On the other hand, the AHN dataset is a real-world dataset with three classes: Unchanged, New Building, and Demolition. When comparing the performance of the three models we will only consider the three common classes between the two datasets, and the observations are as below:

The RF model showed a higher overall recall on the AHN dataset (94.00%) compared to the URB3DCD dataset (93.6%). However, the main reason for this is the big difference in point density and distribution of classes. The unchanged class in AHN dataset has much more points and since "Unchanged" it's the most represented class in both training and validation datasets, it has a very high accuracy, which lead into an artificial increase of overall accuracy of the model. This is why per-class comparison is a very important part of this study.

Table 12 - Validation datasets classes distribution

| Class Name | Class Number | Points in the AHN dataset | Points in the Urb dataset | Urb dataset | AHN dataset |
|--------------|--------------|---------------------------|---------------------------|-------------|-------------|
| Unchanged | 0 | 1008648 | 403061 | 85% | 90% |
| New building | 1 | 62914 | 27266 | 6% | 6% |
| Demolition | 2 | 51681 | 43722 | 9% | 5% |

When applied to the AHN dataset, our model faced considerable challenges, particularly in misclassifying many individual points, with unchanged tree points often mislabelled as building changes. This issue was also present in the simulated dataset, but at a significantly reduced rate. This discrepancy can be attributed to several factors, including the different features used in the two datasets. The value of RGB features has been well established in many classification tasks, while the results from using only intensity, number of returns, and the manually crafted 2D point density feature remain to be fully evaluated.

The state of the data itself also plays a crucial role. In the simulated dataset, tree points are only present on the outer boundaries, so when matched one-to-one between two epochs, they were mostly associated with other tree points, minimizing the likelihood of mismatches with ground point. Our matching process is done in 2D, thus in such scenarios as AHN, the possibility of tree points matching with ground points is high. This is evident in the Figure 55, which clearly shows the higher point density of the AHN dataset, making incorrect point matches more often.

Moreover, it's worth noting that the simulated dataset's "training, testing, and validation" are all situated in the same city (Lyon, France), while our real dataset's training and testing take place in different locations for some of its tiles. The direct comparison of classes, shown in the Figure 54, clearly indicates superior results from the simulated dataset over the real dataset. This suggests that the model's performance is highly sensitive to feature selection, the quality of training data, and how representative the data is.

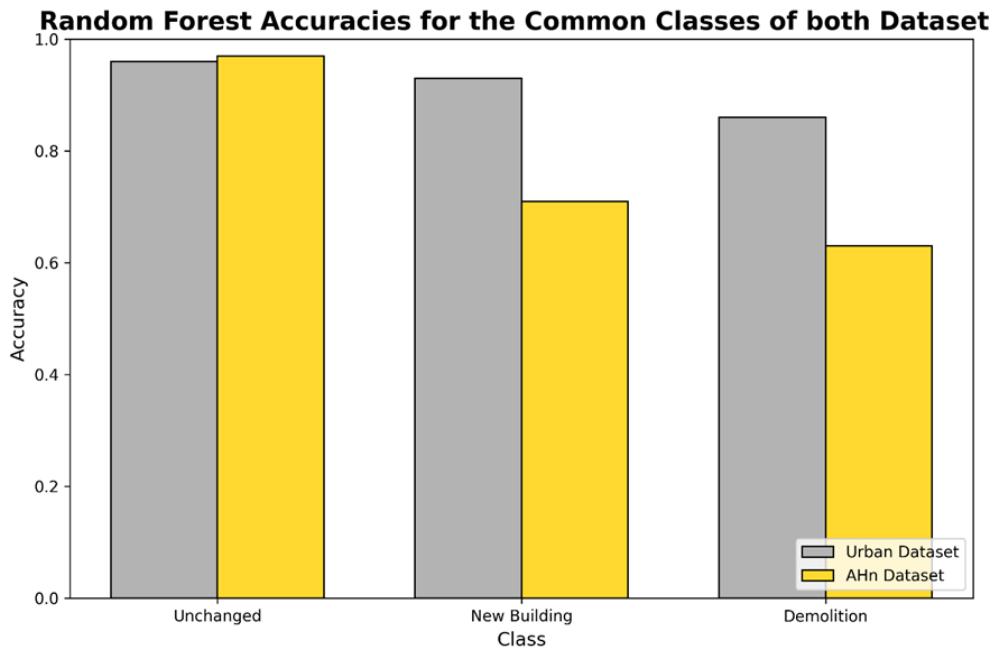


Figure 54 - Comparison of AHN results with Urb3DCD. RF model

The FCNN model also strong performance on the Unchanged class on both datasets. Given that FCNN and RF are trained in a similar fashion, with the only distinction being that FCNN also generates more complex features from the input, the issues addressed in the previous paragraph are relevant here as well. The algorithms showed similar improvements in both the datasets when compared to RF, where they did a better distinction between classes with similar features. The overall performance was better than RF for both the datasets. The per-class comparison between these two models is shown in the Figure 56.

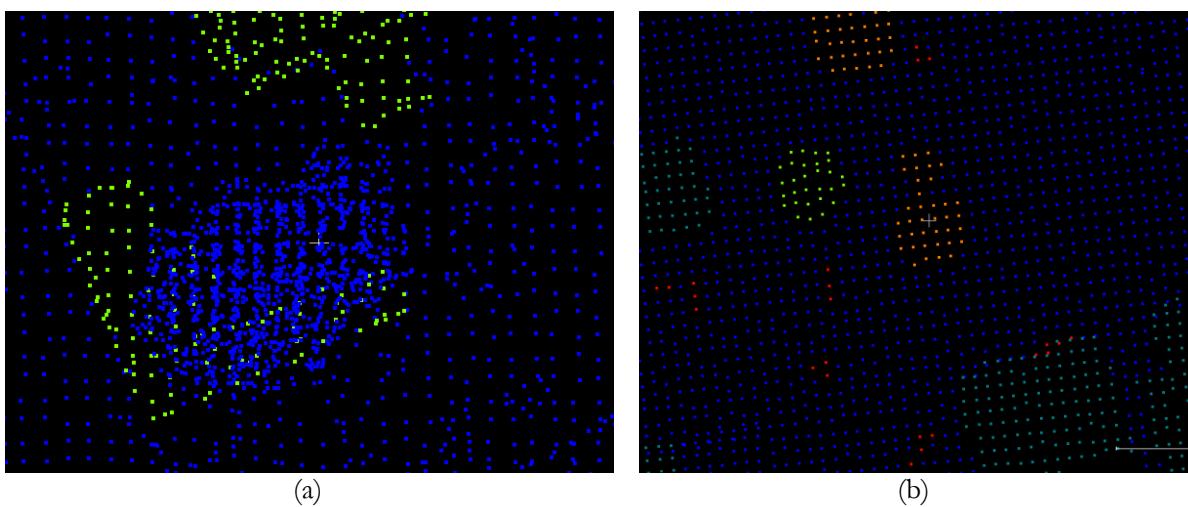


Figure 55 - A closer look at the densities of our datasets. Where (a) is an AHN and (b) a Urb3DCD tile

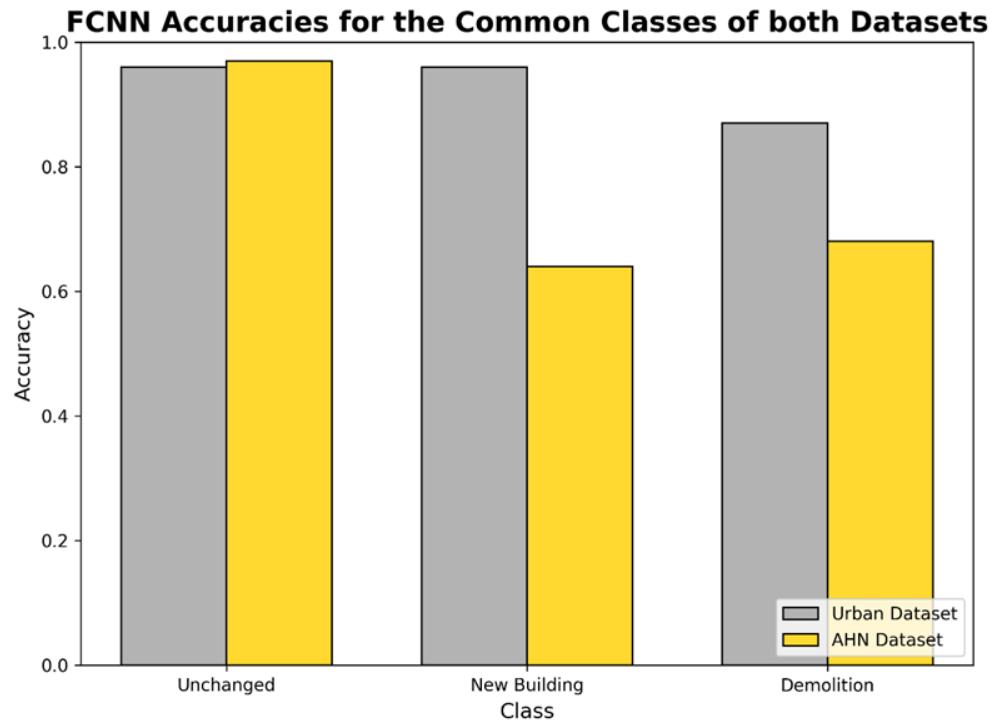


Figure 56 - Comparison of AHN results with Urb3DCD. FCNN model

The CNN model displayed impressive overall accuracy on both datasets. However, a per-class comparison is crucial to fully comprehend the true accuracy. Contrary to the RF and FCNN models, the CNN model did not misclassify trees or buildings in either the AHN or Urb3DCD dataset. This implies that the CNN was more effective in differentiating between natural and built environments compared to the other classifiers. The model combines the advantage of creating higher-level features to distinguish between similar classes and spatial relationships of points. Just like the other models, the same can be said for this model, where Urb3DCD showed an overall better performance.

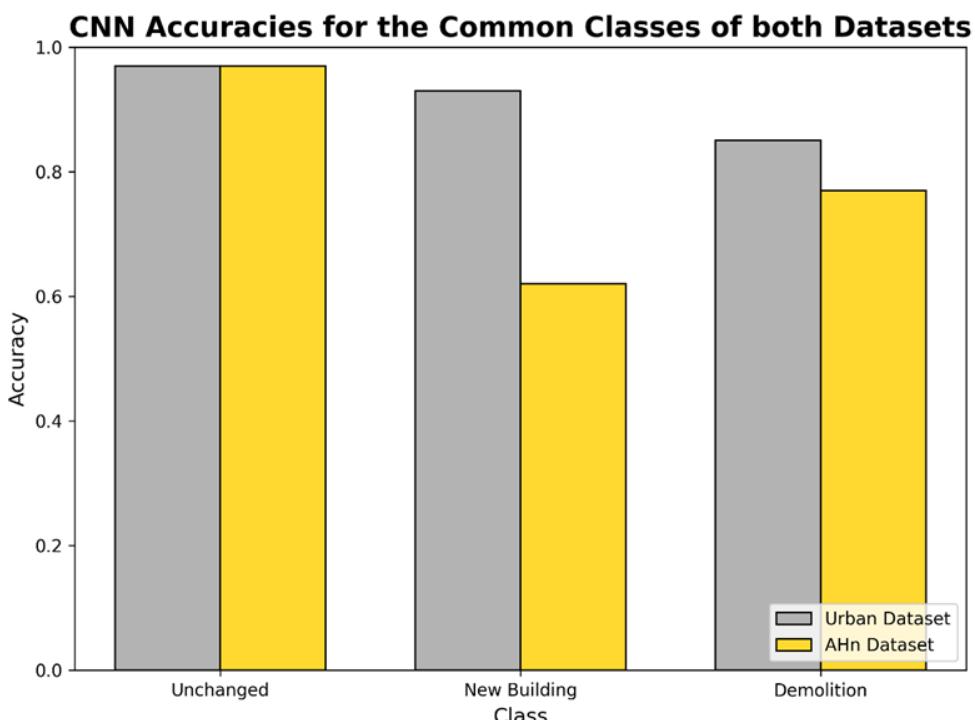


Figure 57 - Comparison of AHN results with Urb3DCD. CNN model

Applying the cluster cleaning to the output of the Urb3DCD dataset wasn't necessary. The simulated dataset's lower density compared to the AHN dataset minimizes the potential for errors. Furthermore, the combination of RGB values and representative of the training data that Urb3DCD offers, led to significantly improved predictions. The quality of the training and validation data for the simulated dataset is far more balanced than the AHN one. Additionally, since the point density of the simulated dataset is much lower, in scenarios where objects have very few points with substantial distance between them, applying cluster cleaning isn't recommended as it can cause problems rather than resolving them.

In conclusion, while all three models (RF, FCNN, and CNN) showed good performance on both datasets, they each demonstrated distinct strengths and weaknesses. The RF model achieved the highest overall accuracy on the AHN dataset but struggled with misclassifications and didn't perform as well on unseen areas. The FCNN model showed robust performance on both datasets but lacked the ability to utilize the spatial relationships between data points. The CNN model excelled in accurately classifying trees and buildings, proving to be the most consistent and reliable algorithm across both datasets. These findings imply that the choice of model and feature set should be carefully considered. A more representative dataset, better feature combination, and consideration of spatial relations are essential for improved prediction. Furthermore, the results highlight the potential of deep learning methods for change detection tasks in urban environments, and the need for further research to optimize these methods for different types of datasets.

6.2. Comparison with the state of the art (Urb3DCD)

A useful perspective on the effectiveness of our method can be obtained by a comparison with other state of the art change detection methods used on the same Urb3DCD dataset. The models developed in this study, Convolutional Neural Networks (CNN), Fully Connected Neural Networks (FCNN), and Random Forests (RF) show a strong balance between ease of use, speed, and performance. Since the previous studies that are tested on the same dataset used the IoU values to evaluate their models, we will match it and the comparison are shown in the Table 13 below:

Table 13 -Per-class IoU scores on Urb3DCD-V2 low density LiDAR dataset. All the other model accuracies are retrieved from de Gelis et al., 2023 study

| Method | Unchanged | New building | Demolition | New veg. | Veg. growth | Missing veg. | Mobile object |
|---|------------------|---------------------|-------------------|------------------|--------------------|---------------------|----------------------|
| Convolution Neural Network (ours) | 94.12 | 81.27 | 77.77 | 36.10 | 50.61 | 63.15 | 99.96 |
| Siamese KPConv (de Gelis et al., 2023) | 95.82 ± 0.48 | 86.68 ± 0.47 | 78.66 ± 0.47 | 93.16 ± 0.27 | 65.17 ± 1.37 | 65.46 ± 0.93 | 91.55 ± 0.60 |
| Pseudo-Siamese KPConv (de Gelis et al., 2023) | 95.20 ± 0.18 | 86.23 ± 1.37 | 76.08 ± 0.54 | 92.98 ± 0.95 | 55.96 ± 9.41 | 63.50 ± 1.41 | 91.88 ± 0.71 |
| DSM-Siamese | 93.21 ± 0.11 | 86.14 ± 0.65 | 69.85 ± 1.46 | 70.69 ± 1.35 | 8.92 ± 15.46 | 60.71 ± 0.74 | 8.14 ± 5.42 |
| DSM-Pseudo-Siamese | 93.44 ± 0.23 | 84.65 ± 2.05 | 68.41 ± 1.77 | 70.38 ± 4.98 | 15.42 ± 13.81 | 59.77 ± 3.32 | 33.15 ± 29.12 |
| DSM-FC-EF | 94.39 ± 0.12 | 91.23 ± 0.31 | 71.15 ± 0.99 | 68.56 ± 3.92 | 1.89 ± 2.82 | 62.34 ± 1.23 | 46.70 ± 3.49 |
| RF (Tran et al. 2018) | 92.72 ± 0.01 | 73.16 ± 0.02 | 64.60 ± 0.06 | 75.17 ± 0.06 | 19.78 ± 0.30 | 7.78 ± 0.02 | 73.71 ± 0.63 |
| Fully Connected Neural Network (ours) | 93.13 | 78.54 | 74.95 | 36.01 | 40.15 | 63.43 | 100.00 |
| Random Forest (ours) | 92.99 | 76.49 | 74.69 | 36.01 | 41.22 | 62.56 | 98.11 |

The CNN model, excels in the 'Unchanged', 'New Building', 'Demolition', and 'Mobile Object' categories, outperforming several other models in these classes, as seen in the Table. While the model shows room for improvement in the, 'Vegetation Growth', and 'Missing Vegetation' classes, it's important to note that it still delivers competitive results, particularly when considering its simplicity and efficiency. The 'New Vegetation' class is the only obvious flaw. On this class, all of our models are underperforming. Looking at the confusion matrix (Figure 36), we encountered difficulties distinguishing between the 'new vegetation' (class 3) and 'vegetation growth' (class 4) categories in our dataset. Despite trying different strategies including up-sampling, weighted loss functions, and ensemble methods such as Random Forests, the Intersection over Union (IoU) metric for class 3 remained poor. First, there might be an overlap in the feature distributions of these two classes, given that both 'new vegetation' and 'vegetation growth' categories represent different stages of a tree's lifecycle, they share similar characteristics, which makes it challenging for our model to effectively differentiate between them. Moreover, our feature set may not enough capture the key differences between these two classes, or they might be missing and are underrepresented in our data. As a result, the model might not have had access to the necessary information to differentiate between these two classes. Lastly, the limitations might also be due to the choice of the model or its architecture. While we used a CNN model, it's possible that other more sophisticated architectures could yield better results. The Siamese KPConv model uses a kernel point convolution operation, which allows it to adapt to the local geometry of the data. This operation is particularly effective for classes like 'Vegetation Growth' that have complex and irregular structures. In contrast, our CNN model, while it does consider neighbouring points, may not capture the complexity of vegetation structures as effectively due to its convolution operation being less adaptable to local geometric variations. Therefore, the adaptability of the Siamese KPConv model's kernel point convolution operation to local geometric variations gives it an edge in identifying and classifying 'Vegetation Growth'. The Siamese KPConv and Pseudo-Siamese KPConv models, as presented by de Gelis et al., 2023, while achieving superior performance in almost all the classes, are more complex and computationally intensive. This highlights the strength of our models, which achieve competitive results with a fraction of the complexity. Similarly, our FCNN and RF models, show very good performance across most classes. The FCNN model, in particular, outperforms both our RF and the Tran et al. 2018 model, demonstrating the potential of neural networks in change detection tasks, even when they are implemented with a simple architecture. The DSM-Siamese, DSM-Pseudo-Siamese, and DSM-FC-EF models, despite their data transformation and pre-processing, did not perform that well. In fact, our CNN model matches or surpasses the performance of the DSM-FC-EF model in several classes. This could be due to the fact that when rasterizing PCs into DSM, the size of the training set is considerably diminished, from one label per point to one label per cell/pixel (a 2D pixel gathers multiple 3D points). Since DSM-based networks lose information and rely on smaller training sets, they might be more prone to over-fitting.

Our models despite their simplicity, have shown competitive performance. They are not only easy to implement but also efficient, as they don't require complex pre-processing steps or extensive training. We demonstrated that simplicity and speed do not have to come at the cost of performance. The Siamese KPConv and Pseudo-Siamese KPConv models offer a sophisticated approach to multiple change segmentation, but their complexity and need for extensive training may not always be necessary. While there are areas for improvement, particularly in the vegetation-related classes, our models deliver competitive results across most classes and outperform most of the other state-of-the-art models in several categories. Our goal was to prove that even by using the idea behind 2D change detection, we can still get very satisfying results in 3D. This way we provide a faster training time as compared to other existing models that deal with raw point cloud data. The simplicity and efficiency of our models make them an attractive choice for change detection tasks, particularly in scenarios where speed and ease of implementation are crucial.

6.3. Comparison with the state of the art (AHN)

Table 14 - Per-class IoU scores on the AHN LiDAR dataset. All the other model accuracies are retrieved from de Gelis et al., 2023 study

| Method | mIoU | Unchanged | New building | Demolition | New clutter |
|---|------------------|------------------|---------------------|-------------------|--------------------|
| CNN (ours) | 66.46 | 93.32 | 52.92 | 53.12 | none |
| Siamese KPConv (de Gelis et al., 2023) | 59.93 ± 0.14 | 95.94 ± 0.06 | 83.19 ± 1.54 | 56.05 ± 1.74 | 40.53 ± 0.56 |
| Pseudo-Siamese KPConv (de Gelis et al., 2023) | 52.32 ± 4.31 | 92.96 ± 1.34 | 76.54 ± 11.39 | 43.67 ± 1.88 | 36.76 ± 2.95 |
| DSM-Siamese | 33.18 ± 3.56 | 88.58 ± 2.53 | 60.95 ± 5.54 | 18.04 ± 1.59 | 20.54 ± 3.59 |
| DSM-Pseudo-Siamese | 41.40 ± 0.62 | 92.25 ± 0.11 | 73.26 ± 0.68 | 22.91 ± 1.82 | 28.02 ± 0.73 |
| DSM-FC-EF | 44.73 ± 2.16 | 92.95 ± 1.49 | 74.21 ± 0.37 | 33.68 ± 6.84 | 26.32 ± 0.04 |
| RF (Tran et al. 2018) | 28.56 ± 0.02 | 93.13 ± 0.00 | 70.5 ± 0.21 | 2.04 ± 0.04 | 13.27 ± 0.02 |
| FCNN (ours) | 65.47 | 93.20 | 56.29 | 46.92 | none |
| RF (ours) | 66.53 | 93.56 | 59.78 | 46.27 | none |

Our comparison with the state of the art also extends to the analysis of the AHN data. However, there are critical points that must be clarified before we dive into the comparison. First of all, it's important to note that our dataset and the datasets used in other state-of-the-art studies differ in both size and geographical location. This discrepancy can cause an unfair comparison, so it's necessary to keep it in mind. Despite these differences, all the datasets share the same fundamental characteristics and sources, being derived from AHN3 and AHN4. Second, as can be observed from the Table 14, our data does not include the class of "new clutter", which will inevitably influence the final mean intersection over union (mIoU) score when placed against other studies. These factors should be kept in mind as this section's comparison and discussion are not as straightforward as those of the simulated dataset.

In terms of individual class performance, our model outperforms all of the competing models, with the exception of Siamese KPConv, in accurately identifying 'Unchanged' and 'Demolition' classes. For instance, our model achieves an 'Unchanged' class accuracy of 93.32%, narrowly trailing the 95.94% of the Siamese KPConv. The majority of other models struggled to identify demolitions on a satisfactory level. Contrarily, our model manages to distinguish 'Demolitions' with an accuracy of 53.12%, surpassing most other models. Our model achieved the highest mean IoU, but it's crucial to underscore that our model does not account for the "new clutter", which will inevitably affect the average accuracy. Therefore, this metric may not be the most reliable indicator of performance, underscoring the importance of per-class comparison. Our model's performance in the 'New Building' class was lower than the other models. Specifically, our model achieved 52.92% accuracy, comparable with the DSM-Siamese results. This low accuracy can be addressed to the data pre-processing step. As described in the methodology section, one substantial challenge was mismatching tree points from one epoch with ground points from the next epoch. We decided to use 2D matching due to its computational speed and effectiveness on the Urb3DCD dataset. While the CNN addressed most of this resulting issues on the AHN dataset, some areas remained where the CNN could not fully fix this error, consequently impacting the accuracy of our classes.

Another challenge that we previously referred to was that our algorithm operates solely on intensity, changes in height, and 2D point density. The importance of RGB values cannot be overstressed in models such as ours, where the individual features of point clouds play a significant role.

Despite the absence of these features, our model still managed to achieve satisfactory results when benchmarked against the current state of the art. These results underscore the potential of our methods, and it lays a solid foundation for further exploration and research in this direction.

7. CONCLUSIONS AND FINAL REMARKS

In this study, we presented a novel deep learning approach to change detection in point clouds, building on the concept of 2D change detection in images and extending it to 3D point clouds. This method was inspired by image change detection research, while also addressing the problems that appear in the traditional change detection and classification methods for 3D data. We developed three models and experimented on two different datasets, one simulated and one real-world. The simulated dataset was generated by (de Gélis et al., 2023), while the real data, derived from official AHN website, was downloaded from Geotiles.nl.

Among the models tested, the CNN showed superior performance due to its utilization of neighbouring points to understand spatial context. The flexibility of our model allows users to easily add or remove classes from training and prediction, enhancing its efficiency. For the simulated dataset, our method achieved state-of-the-art results, as measured by Intersection over Unions (IoU). The CNN model demonstrated very good results across all classes, even surpassing some existing methods, except for the "New Vegetation" class which underperformed across all our models. This underperformance can be attributed to the complexity of vegetation growth class and the limitations of our model in capturing the spatial relationships within this class, as our model does not consider the larger context of these elements.

The real dataset presented more challenges in terms of quality assessment and data preparation due to the absence of some features (such as RGB) and the target labels. Moreover, finding "demolitions" for AHN proved to be difficult, leading us to swap some tile epochs between AHN3 and AHN4 to extract this class from the "new building" one. This artificial crafting of classes might have introduced errors in training and prediction. Additionally, the RGB values, which performed exceptionally well on the Urb3DCD dataset and have shown their value in other classification tasks, were unavailable for our AHN data, leading to a loss in accuracy. Despite these problems, the model performed well when compared to the state of the art.

Point cloud change detection and classification using deep learning methods is a relatively new field, but it has seen significant advancements in recent years. Our objective was to contribute further insights into the potential approaches for dealing with this kind of data. In future work, we recommend integrating RGB features into the AHN data, as these features have demonstrated impressive results in both 2D images and the 3D simulated point cloud. Implementing these features on top of existing ones is expected to boost the model's performance. Furthermore, data inclusivity should be a critical consideration, given the vast diversity in feature combinations for the same object in real-world scenarios. Our models demonstrated strong performance even with reduced training data, as seen in our experiments using models trained with half the dataset. Interestingly, there was minimal difference in performance between full and half dataset experiments, suggesting that representativeness was maintained despite the reduction. Moreover, when we experimented with the AHN data, performance dropped in regions that were quite different from our training areas in terms of feature values. This suggests that the focus should be on the representativeness and quality of the training data, rather than its size. In terms of the optimal amount of training data, it's less about a specific quantity and more about ensuring that the data set includes the range and diversity of the scenarios that we aim to predict. This way we can train robust models that can generalize well and accurately predict new and unseen data.

8. APPENDIX

Random forest results

Table 15 - Accuracies of RF model trained with the first half of the dataset

| Class | Precision | Recall | F1 Score | Support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.941 | 0.963 | 0.952 | 403061 |
| 1 | 0.807 | 0.893 | 0.848 | 27266 |
| 2 | 0.858 | 0.622 | 0.721 | 43722 |
| 3 | 0.38 | 0.882 | 0.531 | 2908 |
| 4 | 0.743 | 0.448 | 0.559 | 9063 |
| 5 | 0.794 | 0.737 | 0.764 | 7306 |
| 6 | 1 | 1 | 1 | 2698 |
| accuracy | | | 0.916 | 496024 |
| macro avg | 0.79 | 0.79 | 0.77 | 496024 |
| weighted avg | 0.92 | 0.92 | 0.91 | 496024 |

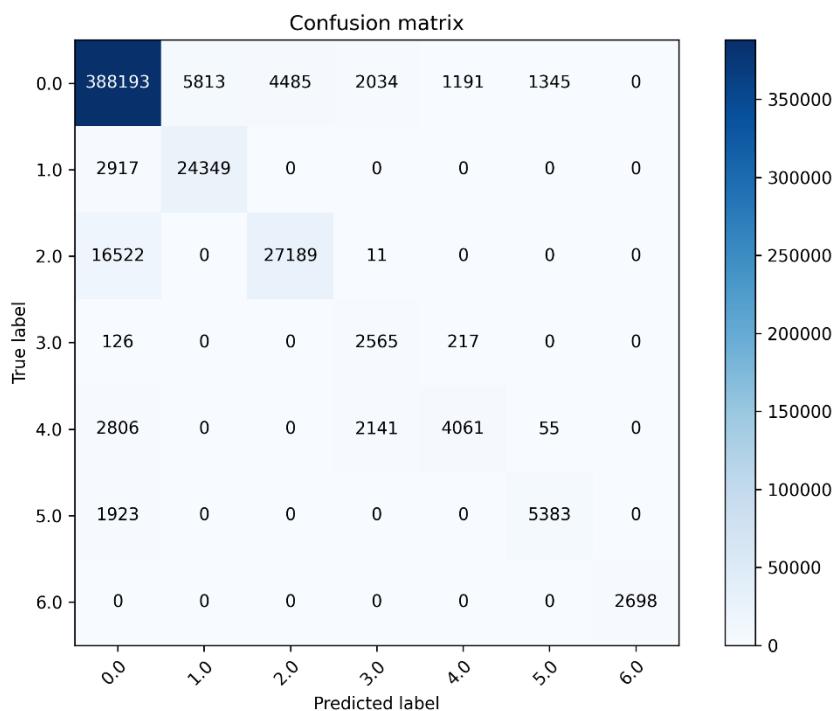


Figure 58 - Confusion matrix of RF model trained on the first half dataset

Table 16 - Accuracies of RF model trained with the second half of the dataset

| Class | Class number | Precision | Recall | F1-score | Support |
|-------------------|--------------|-----------|--------|----------|---------|
| unchanged | 0 | 0.95 | 0.96 | 0.96 | 403061 |
| new_building | 1 | 0.81 | 0.93 | 0.87 | 27266 |
| demolition | 2 | 0.85 | 0.74 | 0.79 | 43722 |
| new_vegetation | 3 | 0.34 | 0.93 | 0.5 | 2908 |
| vegetation_growth | 4 | 0.8 | 0.18 | 0.3 | 9063 |
| vegetation_loss | 5 | 0.8 | 0.74 | 0.77 | 7306 |
| mobile_objects | 6 | 0.93 | 1 | 0.97 | 2698 |
| accuracy | | | | 0.92 | 496024 |
| macro avg | | 0.78 | 0.78 | 0.74 | 496024 |
| weighted avg | | 0.93 | 0.92 | 0.92 | 496024 |

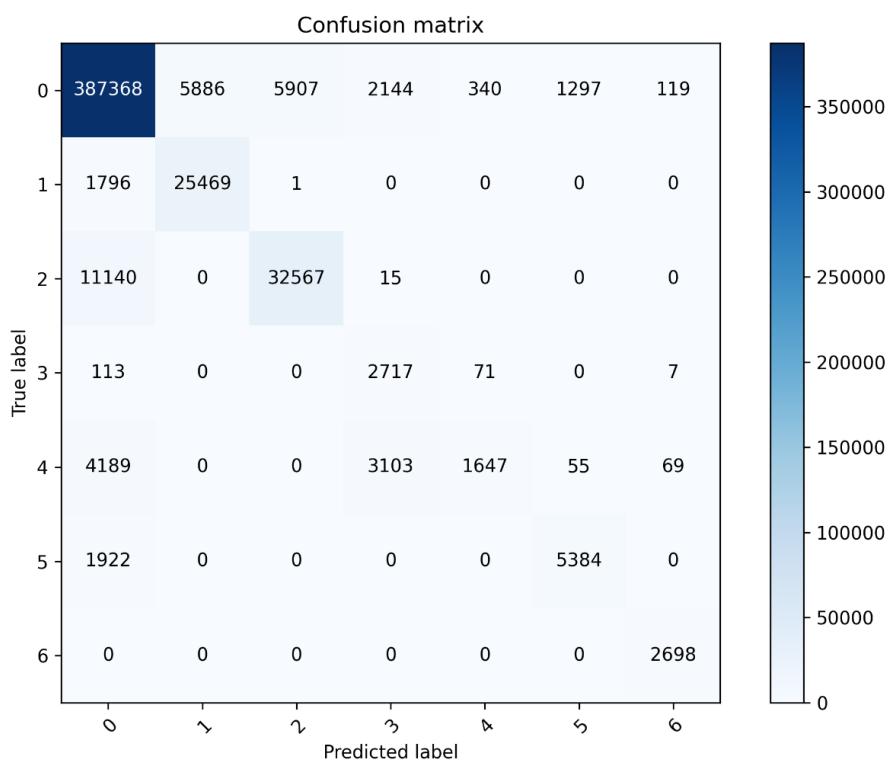


Figure 59 - Confusion matrix of RF model trained on the second half dataset

FCNN results

Table 17 - Accuracies of FCNN model trained with the first half of the dataset

| Class | Class number | Precision | Recall | F1-score | Support |
|-------------------|--------------|-----------|--------|----------|---------|
| unchanged | 0 | 0.968 | 0.959 | 0.963 | 403061 |
| new_building | 1 | 0.812 | 0.925 | 0.865 | 27266 |
| demolition | 2 | 0.847 | 0.863 | 0.855 | 43722 |
| new_vegetation | 3 | 0.371 | 0.926 | 0.53 | 2908 |
| vegetation_growth | 4 | 0.84 | 0.421 | 0.561 | 9063 |
| vegetation_loss | 5 | 0.795 | 0.759 | 0.777 | 7306 |
| mobile_objects | 6 | 1 | 1 | 1 | 2698 |
| accuracy | | | | 0.936 | 496024 |
| macro avg | | 0.805 | 0.836 | 0.793 | 496024 |
| weighted avg | | 0.941 | 0.936 | 0.936 | 496024 |

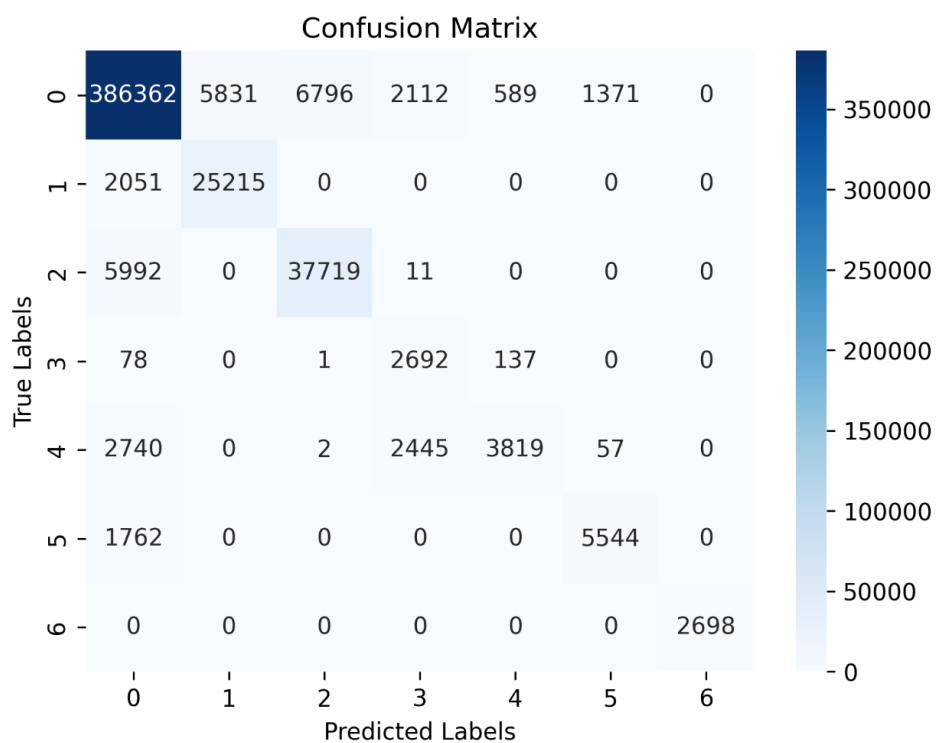


Figure 60 - Confusion matrix of FCNN model trained on the first half dataset

Table 18 - Accuracies of RF model trained with the second half of the dataset

| Class | Class number | Precision | Recall | F1-score | Support |
|-------------------|--------------|-----------|--------|----------|---------|
| unchanged | 0 | 0.96 | 0.96 | 0.96 | 403061 |
| new_building | 1 | 0.81 | 0.94 | 0.87 | 27266 |
| demolition | 2 | 0.86 | 0.82 | 0.84 | 43722 |
| new_vegetation | 3 | 0.37 | 0.93 | 0.53 | 2908 |
| vegetation_growth | 4 | 0.83 | 0.43 | 0.56 | 9063 |
| vegetation_loss | 5 | 0.8 | 0.75 | 0.77 | 7306 |
| mobile_objects | 6 | 1 | 1 | 1 | 2698 |
| accuracy | | | | 0.93 | 496024 |
| macro avg | | 0.81 | 0.83 | 0.79 | 496024 |
| weighted avg | | 0.94 | 0.93 | 0.93 | 496024 |

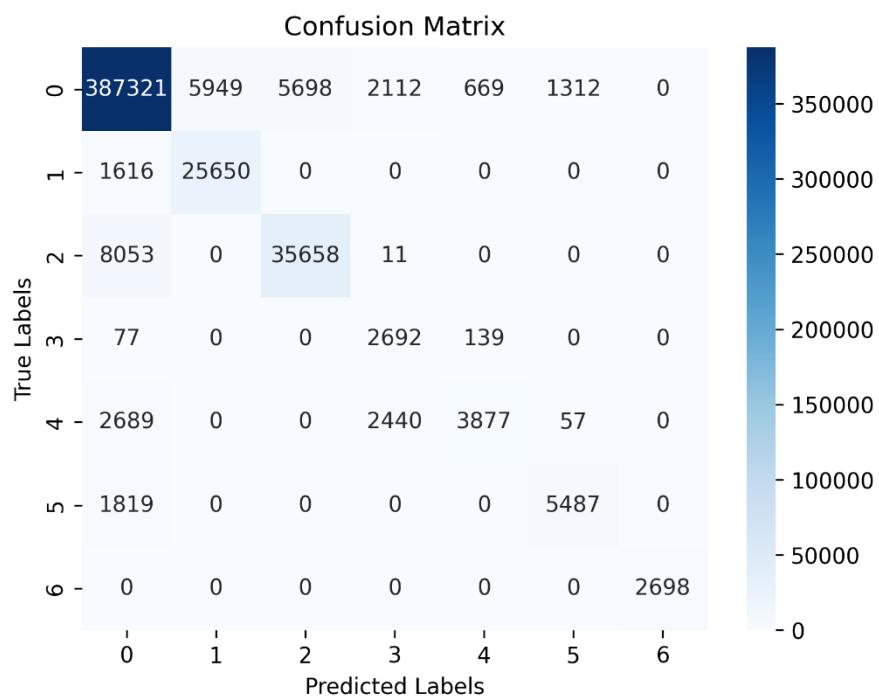


Figure 61 - Confusion matrix of FCNN model trained on the second half dataset

CNN results

Table 19 - Accuracies of CNN model trained with the first half of the dataset

| Class | Class Number | Precision | Recall | F1-score | Support |
|-------------------|--------------|-----------|--------|----------|---------|
| unchanged | 0 | 0.97 | 0.969 | 0.97 | 403061 |
| new_building | 1 | 0.855 | 0.938 | 0.895 | 27266 |
| demolition | 2 | 0.941 | 0.818 | 0.876 | 43722 |
| new_vegetation | 3 | 0.371 | 0.925 | 0.529 | 2908 |
| vegetation_growth | 4 | 0.713 | 0.63 | 0.669 | 9063 |
| vegetation_loss | 5 | 0.779 | 0.775 | 0.777 | 7306 |
| mobile_objects | 6 | 1 | 1 | 1 | 2698 |
| accuracy | | | | 0.945 | 496024 |
| macro avg | | 0.804 | 0.865 | 0.816 | 496024 |
| weighted avg | | 0.95 | 0.945 | 0.946 | 496024 |

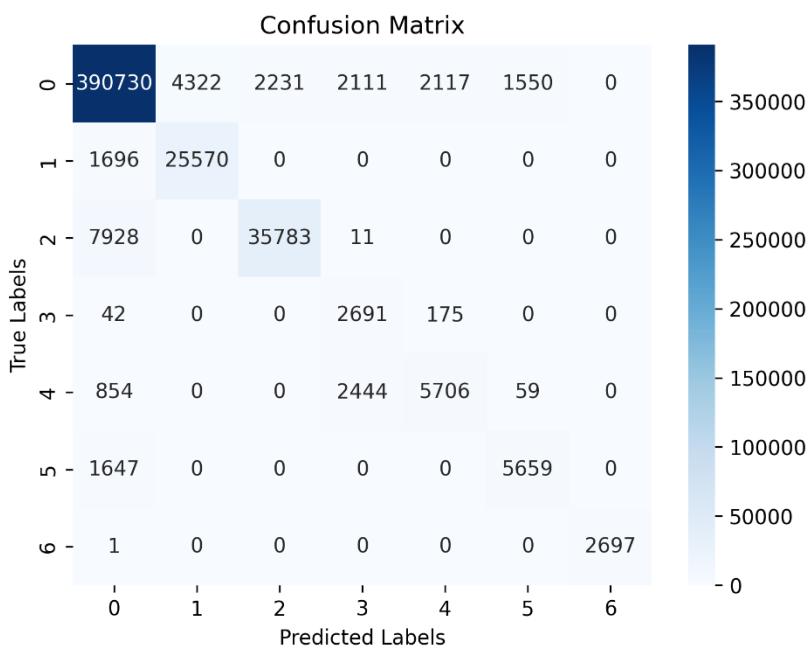


Figure 62 - Confusion matrix of CNN model trained on the first half dataset

Table 20- Accuracies of CNN model trained with the second half of the dataset

| Class | Class Number | Precision | Recall | F1-score | Support |
|-------------------|--------------|-----------|--------|----------|---------|
| unchanged | 0 | 0.964 | 0.977 | 0.971 | 403061 |
| new_building | 1 | 0.87 | 0.926 | 0.897 | 27266 |
| demolition | 2 | 0.955 | 0.807 | 0.875 | 43722 |
| new_vegetation | 3 | 0.37 | 0.927 | 0.529 | 2908 |
| vegetation_growth | 4 | 0.899 | 0.457 | 0.606 | 9063 |
| vegetation_loss | 5 | 0.787 | 0.766 | 0.776 | 7306 |
| mobile_objects | 6 | 1 | 1 | 1 | 2698 |
| accuracy | | | | 0.946 | 496024 |
| macro avg | | 0.835 | 0.837 | 0.808 | 496024 |
| weighted avg | | 0.951 | 0.946 | 0.946 | 496024 |

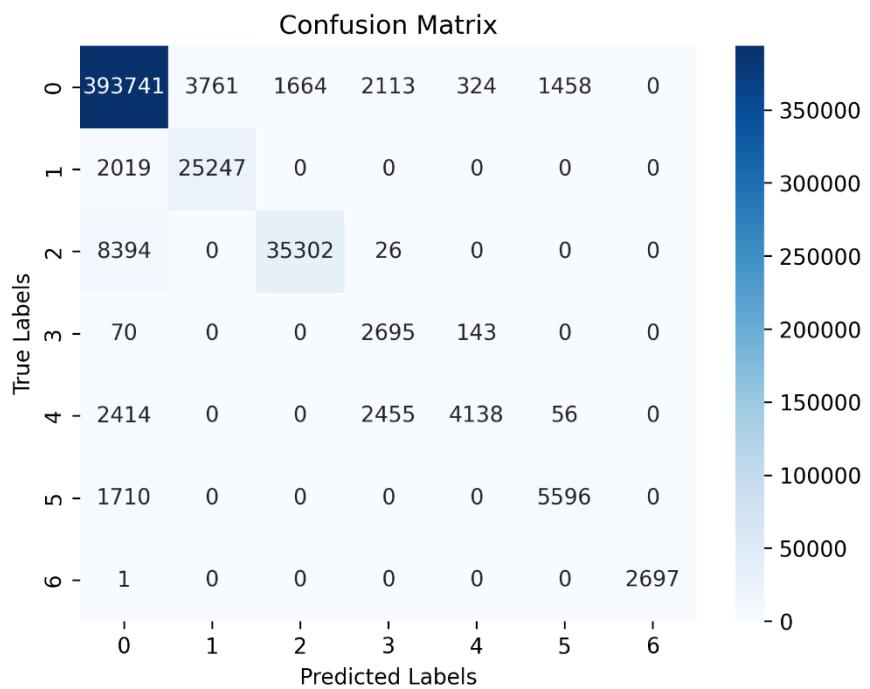


Figure 63 - Confusion matrix of CNN model trained on the second half dataset

LIST OF REFERENCES

- Belgiu, M., & Drăguț, L. (2016). Random forest in remote sensing: A review of applications and future directions. *ISPRS Journal of Photogrammetry and Remote Sensing*, 114, 24–31.
<https://doi.org/10.1016/j.isprsjprs.2016.01.011>
- Blaschke, T. (2010). Object based image analysis for remote sensing. *ISPRS J Photogramm Remote Sens. ISPRS Journal of Photogrammetry and Remote Sensing*, 65, 2–16.
<https://doi.org/10.1016/j.isprsjprs.2009.06.004>
- Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32.
<https://doi.org/10.1023/A:1010933404324>
- Chen, G., Hay, G. J., Carvalho, L. M. T., Wulder, M. A., Chen, G., Hay, G. J., Carvalho, L. M. T., & Wulder, M. A. (2012). *Object-based change detection*. 1161.
<https://doi.org/10.1080/01431161.2011.648285>
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, 2493–2537.
- Cserép, M., & Lindenbergh, R. (2023). Distributed processing of Dutch AHN laser altimetry changes of the built-up area. *International Journal of Applied Earth Observation and Geoinformation*, 116, 103174.
<https://doi.org/10.1016/j.jag.2022.103174>
- Dalwin, D., N., S., & Sathyabama, B. (2018). A novel feature descriptor for automatic change detection in remote sensing images. *The Egyptian Journal of Remote Sensing and Space Science*, 22.
<https://doi.org/10.1016/j.ejrs.2018.03.005>
- de Gélib, I., Lefèvre, S., & Corpetti, T. (2021). Change Detection in Urban Point Clouds: An Experimental Comparison with Simulated 3D Datasets. *Remote Sensing*, 13, 2629.
<https://doi.org/10.3390/rs13132629>
- de Gélib, I., Lefèvre, S., & Corpetti, T. (2023). Siamese KPConv: 3D multiple change detection from raw point clouds using deep learning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 197, 274–291.
<https://doi.org/10.1016/j.isprsjprs.2023.02.001>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.
- Guerin, C., Binet, R., & Deseilligny, M. (2014). Automatic Detection of Elevation Changes by Differential DSM Analysis: Application to Urban Areas. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal Of*, 7, 4020–4037. <https://doi.org/10.1109/JSTARS.2014.2300509>
- Guerin, C., Binet, R., & Pierrot-Deseilligny, M. (2014). Automatic detection of elevation changes by differential DSM analysis: Application to urban areas. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(10), 4020–4037. <https://doi.org/10.1109/JSTARS.2014.2300509>
- Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2019). *Deep Learning for 3D Point Clouds: A Survey (IEEE TPAMI 2020)*.
- Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2020). Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 1.
<https://doi.org/10.1109/TPAMI.2020.3005434>
- He, A., Luo, C., Tian, X., & Zeng, W. (2018). *A Twofold Siamese Network for Real-Time Object Tracking*.
- Hebel, M., Arens, M., & Stilla, U. (2013). Change detection in urban areas by object-based analysis and on-the-fly comparison of multi-view ALS data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 86, 52–64. <https://doi.org/10.1016/j.isprsjprs.2013.09.005>
- Hussain, M., Chen, D., Cheng, A., Wei, H., & Stanley, D. (2013). Change detection from remotely sensed images: From pixel-based to object-based approaches. *International Journal of Photogrammetry and Remote Sensing*, 80, 91–106. <https://doi.org/10.1016/j.isprsjprs.2013.03.006>
- Im, J., & Jensen, J. (2005). A change detection model based on neighborhood correlation image analysis and decision tree classification. *Remote Sensing of Environment*, 99, 326–340.
<https://doi.org/10.1016/j.rse.2005.09.008>
- Jean, S., Cho, K., Memisevic, R., & Bengio, Y. (2014). *On Using Very Large Target Vocabulary for Neural Machine Translation*. 1. <https://doi.org/10.3115/v1/P15-1001>
- Jha, C., & UNNI, N. (1994). Digital change detection of forest conversion of a dry tropical Indian forest region. *REMOTE SENSING*, 15, 2543–2552. <https://doi.org/10.1080/01431169408954265>
- Kalantar, B., Ueda, N., Al-Najjar, H. A. H., & Halin, A. A. (2020). Assessment of convolutional neural network architectures for earthquake-induced building damage detection based on pre-and post-

- event orthophoto images. *Remote Sensing*, 12(21), 1–20. <https://doi.org/10.3390/rs12213529>
- Lague, D., Brodu, N., & Leroux, J. (2013). Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (N-Z). *ISPRS Journal of Photogrammetry and Remote Sensing*, 82. <https://doi.org/10.1016/j.isprsjprs.2013.04.009>
- Lehtola, V., Kaartinen, H., Nuchter, A., Kaijaluoto, R., Kukko, A., Litkey, P., Honkavaara, E., Rosnell, T., Vaaja, M., Virtanen, J.-P., Kurkela, M., Issaoui, A., Zhu, L., Jaakkola, A., & Hyppä, J. (2017). Comparison of the Selected State-Of-The-Art 3D Indoor Scanning and Point Cloud Generation Methods. *Remote Sensing*, 9, 796. <https://doi.org/10.3390/rs9080796>
- Liu, K., Boehm, J., & Alis, C. (2016). Change detection of mobile LIDAR data using cloud computing. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLI-B3, 309–313. <https://doi.org/10.5194/isprs-archives-XLI-B3-309-2016>
- Liu, M., Chai, Z., Deng, H., & Liu, R. (2022). A CNN-Transformer Network With Multiscale Context Aggregation for Fine-Grained Cropland Change Detection. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15, 1. <https://doi.org/10.1109/JSTARS.2022.3177235>
- Liu, W., Sun, J., Li, W., Hu, T., & Wang, P. (2019). Deep Learning on Point Clouds and Its Application: A Survey. *Sensors*, 19, 4188. <https://doi.org/10.3390/s19194188>
- Liu, X., & Jr, R. (2002). Urban change detection based on an artificial neural network. *International Journal of Remote Sensing - INT J REMOTE SENS*, 23, 2513–2518. <https://doi.org/10.1080/01431160110097240>
- Longbotham, N., Pacifici, F., Glenn, T., Zare, A., Volpi, M., Tuia, D., Christophe, E., Michel, J., Inglada, J., Chanussot, J., & Du, Q. (2012). Multi-Modal Change Detection, Application to the Detection of Flooded Areas: Outcome of the 2009–2010 Data Fusion Contest. *IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING*, 1. <https://doi.org/10.1109/JSTARS.2011.2179638>
- Lyon, J., Yuan, D., Lunetta, R., & Elvidge, C. (1998). A Change Detection Experiment Using Vegetation Indices. *Photogrammetric Engineering and Remote Sensing*, 64.
- Manavalan, P., Kesavasamy, K., & Adiga, S. (1995). Irrigated crops monitoring through seasons using digital change detection analysis of IRS-LISS 2 data. *International Journal of Remote Sensing - INT J REMOTE SENS*, 16, 633–640. <https://doi.org/10.1080/01431169508954430>
- Muchoney, D., & Haack, B. (1994). Change detection for monitoring forest defoliation. *Photogrammetric Engineering and Remote Sensing*, 60, 1243–1251.
- Nelson, R. (1983). Detecting forest canopy change due to insect activity using Landsat MSS. *Photogrammetric Engineering and Remote Sensing*, 49.
- Nemoto, K., Hamaguchi, R., Sato, M., Fujita, A., Imaizumi, T., & Hikosaka, S. (2017). Building change detection via a combination of CNNs using only RGB aerial imageries. *Remote Sensing Technologies and Applications in Urban Environments II*, 10431, 107–118.
- O, Y. (1998). Principal component analysis of stacked multi-temporal images for the monitoring of rapid urban expansion in the Pearl River Delta. *International Journal of Remote Sensing*, 19, 1501–1518. <https://doi.org/10.1080/014311698215315>
- Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua*(January 2017), 77–85. <https://doi.org/10.1109/CVPR.2017.16>
- Remondino, F., Spera, M., Nocerino, E., Menna, F., & Nex, F. (2014). State of the art in high density image matching. *The Photogrammetric Record*, 29, 144–166. <https://doi.org/10.1111/phor.12063>
- Singh, A. (1989). Review Article: Digital change detection techniques using remotely-sensed data. *International Journal of Remote Sensing*, 10(6), 989–1003. <https://doi.org/10.1080/01431168908903939>
- Sohl, T. (1999). Change analysis in the United Arab Emirates: An investigation of techniques. *Photogrammetric Engineering and Remote Sensing*, 65, 475–484.
- Stal, C., Tack, F., De Maeyer, P., Wulf, A., & Goossens, R. (2013). Airborne photogrammetry and LIDAR for DSM extraction and 3D change detection over an urban area: a comparative study. *International Journal of Remote Sensing*, 34, 1087–1110. <https://doi.org/10.1080/01431161.2012.717183>
- Still, U., & Xu, Y. (2023). *Change detection of urban objects using 3D point clouds: A review*. 197, 228–255. <https://doi.org/10.1016/j.isprsjprs.2023.01.010>
- Thomas, H., Ruizhongtai Qi, C., Deschaud, J.-E., Marcotegui, B., Goulette, F., & Guibas, L. (2019). *KPConv: Flexible and Deformable Convolution for Point Clouds*.
- Tran, T. H. G., Ressl, C., & Pfeifer, N. (2018). Integrated change detection and classification in urban areas based on airborne laser scanning point clouds. *Sensors (Switzerland)*, 18(2).

- <https://doi.org/10.3390/s18020448>
- United Nations. (2008). Spatial Planning - Key Instrument for Development and Effective Governance with Special Reference to Countries in Transition. *Economic Commission for Europe, March*, 1–56.
- Vosselman, G., & Maas, H.-G. (2010). *Airborne and Terrestrial Laser Scanning*.
- Woodcock, C., Macomber, S., Pax-Lenney, M., & Cohen, W. (2001). Monitoring large areas for forest change using Landsat: Generalization across space, time and Landsat sensors. *Remote Sensing of Environment*, 78. [https://doi.org/10.1016/S0034-4257\(01\)00259-0](https://doi.org/10.1016/S0034-4257(01)00259-0)
- Xiao, J., Adler, B., & Zhang, H. (2012). *3D point cloud registration based on planar surfaces*.
<https://doi.org/10.1109/MFI.2012.6343035>
- Xu, H., Cheng, L., Li, M., Chen, Y., & Zhong, L. (2015). Using octrees to detect changes to buildings and trees in the urban environment from airborne LiDAR data. *Remote Sensing*, 7(8), 9682–9704.
<https://doi.org/10.3390/rs70809682>
- Yadav, R., Nascetti, A., & Ban, Y. (2022). Building Change Detection Using Multi-Temporal Airborne Lidar Data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, 43(B3-2022), 1377–1383. <https://doi.org/10.5194/isprs-archives-XLIII-B3-2022-1377-2022>
- Yastikli, N., & Cetin, Z. (2021). Classification of raw LiDAR point cloud using point-based methods with spatial features for 3D building reconstruction. *Arabian Journal of Geosciences*, 14(3).
<https://doi.org/10.1007/s12517-020-06377-5>
- Yotov, K., Hadzhikolev, E., Hadzhikoleva, S., & Cheresharov, S. (2023). Finding the Optimal Topology of an Approximating Neural Network. *Mathematics*, 11, 217. <https://doi.org/10.3390/math11010217>
- Zhan, Y., Fu, K., Yan, M., Sun, X., Wang, H., & Qiu, X. (2017). Change Detection Based on Deep Siamese Convolutional Network for Optical Aerial Images. *IEEE Geoscience and Remote Sensing Letters*, 14(10), 1845–1849. <https://doi.org/10.1109/LGRS.2017.2738149>