POLITECNICO DI TORINO

POLITECNICO DI TORINO

Master Degree course in Artificial Intelligence And Data Analytics
Computer Engineering

Master Degree Thesis

Leveraging AI for the Analysis of Historical Monuments and the Processing of Cultural Heritage Data

Supervisor
Prof. Andrea Bottino

Candidate
Michele Pulvirenti

Academic Year 2023-2025

Acknowledgements

In queste righe voglio esprimere i miei ringraziamenti a tutte le persone che hanno reso possibile questo percorso e che mi hanno supportato in vari modi.

Vorrei innanzitutto ringraziare la mia famiglia per il supporto e l'incoraggiamento che mi hanno sempre dato durante i miei studi. In particolare, un ringraziamento speciale ai miei genitori che mi hanno dato gli strumenti per vivere questa avventura.

Un grazie a mia madre per avermi sempre incoraggiato anche quando perdevo la voglia di studiare. Tra una chiaccherata in pausa caffé e, a distanza, una videochiamata mi ha sempre fatto sentire che ero sulla strada giusta.

Un grazie a mio padre per avermi sempre aiutato e spinto a dare il meglio di me. Mi ha sempre sostenuto e fatto sentire che potevo raggiungere qualsiasi obiettivo.

Vorrei ringraziare anche i miei amici e tutte le persone conosciute durante questo percorso universitario con cui ho condiviso gioie, emozioni e dolori.

A tutte le persone con cui sono cresciuto ad Acireale (e dintorni) che, ogni volta che torno, mi fanno sempre sentire come se non fossi mai andato via e che ho sempre il piacere di rivedere. Di loro porto sempre con me i ricordi piú belli e profondi.

A tutte le persone conosciute a Torino con cui ho condiviso una parte importante della mia vita e con cui nascono sempre progetti e idee. Insieme a loro ho sempre sentito Torino come una seconda casa.

Infine, a tutte le persone conosciute a Parigi con cui ho condiviso quest'ultimo anno e con cui ho ampliato la mia mente a nuovi orizzonti. Grazie per aver reso questa esperienza indimenticabile.

A voi e a tutti quelli che mi sono vicini in questi giorni, un grazie di cuore.

Abstract

This thesis presents the research and development of Machine Learning and AI techniques for the analysis and processing of cultural heritage data.

The main objective is to design a point cloud segmentation application for cultural monuments, addressing the challenges posed by large and heterogeneous data.

Two approaches are explored: a machine learning classifier trained on a partially segmented point cloud to label the remaining points, and neural network models trained on annotated datasets to segment points into predefined classes.

Despite limitations caused by the scarcity of labeled data, both methods proved effective and are integrated into a user-friendly interface, enabling users to load point clouds, select an approach, adjust parameters, and execute the segmentation.

Additionally, a state-of-the-art survey on text recognition in handwritten manuscripts is conducted to assess current AI/ML tools for extracting and organizing text based on relevant keywords. Although this work remained in the research phase, it provided valuable insights into existing technologies and future directions.

In general, this work combines applied machine learning development, exploration of emerging technologies, and integration of practical tools, contributing to the digitization and processing of cultural heritage assets.

Contents

Chapter 1

Introduction

1.1 Overview

This work was completed during my six-month internship as part of the one-year Erasmus program in Paris. The main objective was to exploit Machine Learning and AI techniques to strengthen the company tools for analyzing and processing heritage data.

In the company, I occupied the position of "Intern Engineer" focused on ML/AI solutions.

1.2 The company

A-BIME is a design and expertise firm specializing in technical assistance for the restoration and preservation of historical buildings and monuments.

Figure 1.1: Company logo

Founded in 2015, A-BIME serves a variety of clients in the construction and heritage sector, including project owners, architects, construction companies, and property agencies. A-BIME uses a unique methodology that involves transcribing archival documents and heterogeneous data sources into a digital model, aligning with the Building Information Modeling (BIM) approach.

Main company's services include:

- Diagnostics for historic buildings (structure, pathology, termal, energetical, historical, archeological, lightning and more)
- HBIM (Heritage BIM) modeling
- Virtual tours, lifecycle management and maintenance support

- Development of digital solution for maintenance, management and conservation of historical monuments

1.2.1 Corporate social responsibility policy

Beyond its design of BIM, A-BIME is committed to a strong approach of environmental, social, and digital certification.

A-BIME embodies, through its primary mission, some of these social considerations with the deep conviction that knowledge allows for a better understanding of the built heritage and thus to preserve it over time while respecting its historical authenticity. The decisions of project owners, owners or managers are thus made in an informed and objective manner in a necessary and effective continuity to meet current challenges.

A-BIME actively participates in studies conducted by various institutions and in the reflections of working groups on the preservation, conservation and restoration of ancient heritage related to environmental, energy and societal issues.

Its corporate social responsibility policy is organized around 3 main axes:

- Social and environmental responsibility (CSR)
- Digital Corporate Responsibility
- Cultural and heritage responsibility

For each strategic axis, A-BIME relies on recognized labels and commitments, created by public or private bodies but validated by public authorities.

1.3 Addressed scenario and problem

Processes such as point cloud segmentation and manuscript text extraction can be extremely time-consuming, requiring significant manual effort and attention to detail.

By the automation of these tasks, we can enhance efficiency and streamline workflows, allowing professionals to focus on more critical aspects of their work.

While several research studies have already been conducted on various objectives related to these areas, this thesis is dedicated to building practical, implementable solutions that address the challenges associated with these processes.

The goal is to create tools that improve the performances but also make these technologies accessible and user-friendly for professionals in the field.

1.4 Procedure

The work combines literature review, software development and experiments on real data. First, I reviewed academic and applied work on point cloud processing and on handwritten text recognition. Then I implemented two segmentation pipelines: one classifier-based using partial labels, and one neural-network-based using a labelled dataset. I ran experiments to validate that both approaches work in practice. Where possible, I evaluated the output visually and by comparing against available labelled samples. Finally, I packaged the methods in a prototype UI and documented the design choices and limitations.

Chapter 2

Objectives

This work aimed to strengthen the internal digital tools of the company and to participate in the development of processes and methods. This subject falls within the company's research and development program.

2.1 Projects

2.1.1 Point Cloud Segmentation

The primary objective was to develop a computer program capable of segmenting point clouds of historical buildings and monuments. This task was motivated by the need to simplify and automate the analysis of complex 3D scans, which are often large, unstructured, and difficult to manipulate manually.

The program was designed to be user-friendly, allowing users to load a point cloud, perform segmentation, and show or save the resulting data for further analysis. Two segmentation approaches were implemented in the tool, giving users the option to choose the most suitable approach for their task and adjust key parameters.

2.1.2 Manuscripts Handwritten Text Extraction

Another objective was to explore advanced machine learning and artificial intelligence techniques for text recognition in historical manuscripts. The goal was to find solutions capable of automatically extracting textual information from scanned documents and organizing that information based on relevant keywords.

In this work, I performed a comprehensive review of existing tools, datasets, frameworks, and machine learning models suitable for this tasks. Although no implementation was carried out, this research provided a solid foundation for potential future development, helping the company understand the current capabilities and requirements of automated document and image analysis.

2.2 Methods and Tools

2.2.1 Point Cloud Segmentation Tool

All algorithms for the segmentation project were implemented in *Python*. The following tools and libraries were essential to the development:

- *PyTorch* for building and training the neural network models.
- *scikit-learn* for classical machine learning utilities such as preprocessing and evaluation functions.
- *torchscript* to optimize, save and load the trained models efficiently.
- *open3d*, *laspy*, *e57*, and *matplotlib* for loading, visualizing, and processing point cloud data in various formats.
- *PyQt6* for designing a lightweight graphical user interface, enabling non-expert users to easily load a point cloud, choose the segmentation approach, and export the results.
- *numba* to accelerate computationally intensive functions.
- *scipy*, in particular spatial data structures (e.g., KD-trees), for efficient neighborhood queries within large point clouds.

Training and testing were performed on cloud platforms (Google Colab Pro and Kaggle) to take advantage of GPU acceleration, allowing faster experimentation with model architectures and parameters.

2.2.2 Manuscript Text Recognition

For this project, I conducted state-of-the-art research and prepared a detailed report summarizing my findings. I also experimented with the open-source tool *eScriptorium*, installing it through its Docker image, to evaluate its capabilities for extracting text from historical documents.

2.3 Regulatory Aspects

All projects were conducted in compliance with the company's data handling policies and privacy regulations, ensuring that no copyrighted material was improperly used.

Chapter 3

Methodology

3.1 Point Cloud Segmentation

The goal of this project was to create a user-friendly computer program that can perform segmentation on point clouds of heritage buildings.

The project started with an initial phase of research to learn about:

- The basics of point cloud processing in Python
- Existing model architectures and use cases
- Similar research done in the past

3.1.1 Dataset

Figure 3.1: ArCH Dataset Logo

The dataset used is ArCH (Architectural Cultural Heritage), which consists of 17 annotated point clouds, produced notably by the Polytechnic University of Turin in collaboration with other universities.

Many of the buildings included in the ArCH dataset are either part of, or candidates for, the UNESCO World Heritage List (WHL).

A total of nine semantic classes were defined in the dataset, together with an additional class labeled as `"other"`. The latter groups all elements not included in the main categories, such as paintings, altars, benches, statues, or waterspouts.

In the Figure 3.2, we can see an example of an annotated point cloud from the dataset.

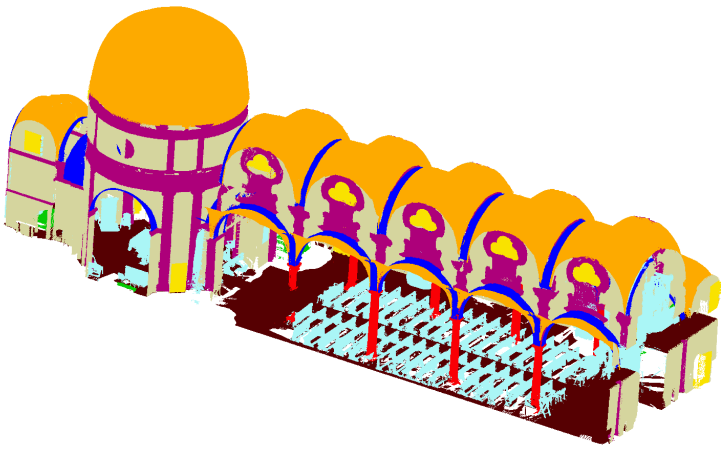

Figure 3.2: Annotated church point cloud example from dataset

The complete list of classes is the following (with corresponding label index):
arch: 0, column: 1, moldings: 2, floor: 3, door_window: 4,
wall: 5, stairs: 6, vault: 7, roof: 8, other: 9.

3.1.2 State of the art

Semantic segmentation of 3D data can generally be approached through three principal methodologies:

- Projection-based methods: These techniques transform 3D point clouds into 2D representations (e.g., RGB images), enabling the use of 2D CNNs. SAM3D [16] extends this idea by projecting 2D masks from SAM onto 3D point clouds for zero-shot 3D segmentation.
- Voxel-based methods: In this approach, the spatial domain is discretized into volumetric grids that can be processed by three-dimensional convolutional neural networks (3D CNNs). Despite their effectiveness in certain contexts, voxel-based methods are less frequently used in cultural heritage applications because of their limited ability to preserve fine geometric details.
- Point-based methods: These methods operate directly on raw point cloud coordinates, avoiding the need for handcrafted feature extraction. Point-based techniques have emerged as the prevailing paradigm in contemporary point cloud segmentation research.

Academic Research Approaches

Early heritage segmentation used classical ML techniques on hand-crafted geometric features such as covariance-based features on spherical neighborhoods [8].

These supervised ML approaches (feature engineering + classifier) struggled to generalize due to the variety of architectural styles and details.

The advent of deep learning revolutionized point-cloud segmentation. Modern methods include:

- PointNets: PointNet [11] and PointNet++ [12] were among the first networks to feed raw point coordinates through neural networks. They capture global and local shape but have limited ability to model a fine-grained context by themselves.
- Graph-Based CNNs: DGCNN [15] constructs a dynamic k-NN graph and applies EdgeConv to learn local patterns. These networks can diffuse information across the point sets and have become very popular in heritage research.
- Heritage-specific adaptations: Researchers often augment the networks with domain features. Pierdicca et al. (2020) [10], for example, applied DGCNN to ArCH by adding HSV color and normal vectors as input features. Matrone et al. (2020) [9] added geometric features such as planarity and verticality and compared various classifiers on ArCH, improving accuracy. Other strategies include data augmentation and transfer learning. These efforts highlight that standard network architectures often need heritage-specific tailoring.

3.1.3 Models

Two model architectures were selected for this task: PointNet++ [12] and DGCNN [15].

PointNet++

PointNet++ is a deep learning architecture for 3D point clouds that extends PointNet with a hierarchical structure. It samples points using Farthest Point Sampling (FPS), groups neighbors via radius or k-nearest neighbors, and applies a mini-PointNet to each local group. By repeating this across layers, the model captures both local and global geometric features effectively.

DGCNN

DGCNN (Dynamic Graph Convolutional Neural Network) processes point clouds using a dynamic k-nearest neighbor graph at each layer. Its EdgeConv operation computes edge features by combining each point's features with its neighbors, updating the graph iteratively. This allows DGCNN to capture fine-grained local geometry and relational structures efficiently.

Modified versions

In order to increase model performance and to better handle the data, PointNet++ was adapted by explicitly including XYZ coordinates, combining Multi-Scale and single-scale grouping, and adding a Transformer layer for contextual information. Residual connections were introduced in both feature propagation and the segmentation head to enhance stability.

DGCNN was modified by integrating residual EdgeConv blocks, concatenating local and global features, and employing a deeper segmentation head with Conv-GroupNorm-LeakyReLU blocks and dropout for improved fine-grained segmentation.

3.1.4 Challenges and Open Problems

Despite rapid progress, key gaps remain.

A major issue is data scarcity: ArCH (17 labeled scans) is one of the very few datasets explicitly for heritage. Most segmentation models are trained on small, domain-specific datasets and may not generalize across eras, styles, or sensor modalities. Complex architectural details (carvings, ornaments) and damage (erosion) can cause over-segmentation or misclassification.

A recent work has shown that synthetic point clouds generated through diffusion-based methods can further improve model performance by enriching training datasets with realistic, diverse samples [17].

This approach is particularly helpful in architectural and construction applications, where obtaining large volumes of labeled point cloud data is often impractical. By leveraging synthetic data, models can achieve better generalization and strength in segmenting complex environments.

Deep models need large annotated sets, but expert annotations are costly. Handling multi-source data is also a problem: fusing UAV photogrammetry with terrestrial LiDAR (varying density, color vs intensity data) can confuse models.

In summary, while deep learning has enabled impressive results on structured parts of heritage scans, broad automation of full-building segmentation still requires more research, from better algorithms to richer annotated datasets.

3.1.5 Segmentation pipeline

After the initial research phase, two segmentation approaches were developed and implemented in the final program.

The final pipeline, in order to perform the segmentation, follows common steps in both approaches:

- Point cloud loading
- Voxelization
- Segmentation
- Noise reduction
- De-voxelization

The point cloud is voxelized in order to reduce the computational cost of the data processing and to reduce noise.

After the segmentation, to clean the possibly generated noise, points are taken in small neighboring groups and their label is set to the most frequent one inside the group through majority voting.

Finally, the voxelized point cloud is restored to its original number of points.

3.1.6 Machine Learning approach

The approach

This solution aims to train a classifier on partially segmented data to complete the segmentation.

As mentioned earlier, the point cloud to segment must have an already segmented portion; this part of the data can be manually labelled or an automated method can be used to generate the labels.

Then, the already segmented points are used to train several Random Forest and Extra Trees classifiers, varying their parameters. To do that, the segmented points are split into a training and a validation set.

The Extra Trees classifier belongs, like the Random Forest one, to the ensemble learning family (specifically, bagging of decision trees), but they differ in how they build their trees and inject randomness.

The decision trees are ML models that recursively splits the data based on feature thresholds to create a tree structure where each leaf node represents a class label. They are easy to interpret and can handle both numerical and categorical data, but they can overfit if not properly pruned or regularized.

With noisy data, Extra Trees can sometimes outperform Random Forest. This step is done because we don't know in advance the input data characteristics, and we want to be able to adapt to different scenarios.

The best performing model, the one that achieves the highest IoU (Intersection over Union) metric, is finally selected and used to segment the points that are not yet classified.

Data pre-processing

The entire point cloud is first voxelized to reduce the computational weight of the task.

Each point in a point cloud typically includes three types of basic features: spatial coordinates (XYZ), RGB color values, and surface normals.

If the normals are not provided in the input data, they are estimated using local neighborhoods.

To improve the effectiveness of color information, the RGB values are converted to HSV format. HSV separates color into hue, saturation, and value, making each channel independent and more suitable for feature extraction, as suggested in [10].

In addition to these basic features, several geometric descriptors ([8]) are calculated for each point to capture local shape properties:

- **Curvature:** Measures how much the surface bends at a point, helping to distinguish flat regions from edges or corners.
- **Planarity:** Quantifies how closely the neighboring points fit a plane, useful for identifying planar surfaces.

- **Omnivariance:** Represents the overall variance of the local neighborhood, indicating the complexity of the surrounding geometry.
- **Verticality:** Assesses the orientation of the surface's normal relative to the vertical axis, aiding in the identification of vertical structures.

These features are first scaled to improve convergence and prevent dominance of the features and then PCA is applied to take the most discriminative directions.

Segmentation

Several combinations of parameters are used to train multiple Random Forest and ExtraTrees models, and the one with the highest IoU score is selected.

Then, the model is used to segment the points that are not already classified.

The segmentation relies on the point features, some of them computed considering the neighboring points.

Finally, the point cloud is de-voxelized, restoring the original number of points.

Results and considerations

As already said, this approach requires that a portion of the point cloud is already pre-segmented.

The most reliable way to have this scenario is through manual labeling with external software, although this step can be time-consuming. Manual segmentation ensures high-quality labels, which are crucial for training a strong classifier.

Alternatively, initial segmentation can be performed using automated or semi-automated methods, such as clustering algorithms or rule-based filters, to quickly label large portions of the point cloud. However, these methods may introduce errors or inconsistencies, so manual correction is often necessary to refine the labels.

Once a subset of points is labeled, the method remains flexible, as it does not impose constraints on the number or type of classes to be segmented. The classifier can be trained on any set of classes present in the labeled data, adapting to different architectural styles or segmentation requirements. This flexibility allows the approach to be applied to a wide range of heritage buildings and point cloud datasets, regardless of their complexity or the specific elements to be identified.

In summary, while the initial segmentation step is essential and may require manual effort, it enables the use of machine learning to efficiently propagate labels to the rest of the point cloud, significantly reducing the overall annotation workload and improving segmentation consistency.

But, if some elements are not segmented, the classifier will not be able to recognize them, and it will classify their points as the most similar ones present in the already segmented part.

This limitation arises because the classifier learns only from the classes represented in the initial labeled subset without any other information.

Therefore, the completeness and diversity of the initial segmentation are crucial for the classifier's ability to generalize and accurately propagate labels. To achieve reliable

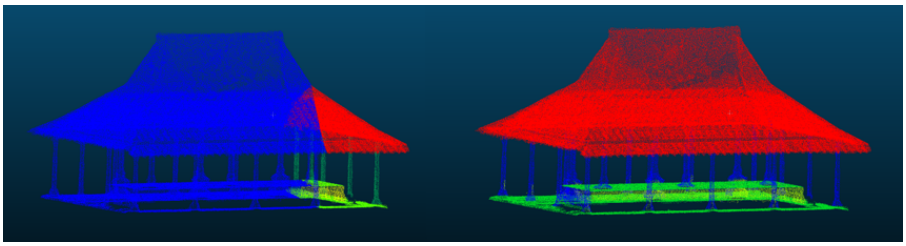

Figure 3.3: Example with Pavillon

segmentation across all desired classes, it is important to ensure that the initial labeled subset contains representative samples of each class present in the point cloud. Otherwise, the classifier will be inherently limited and unable to identify elements that were not included in its training data.

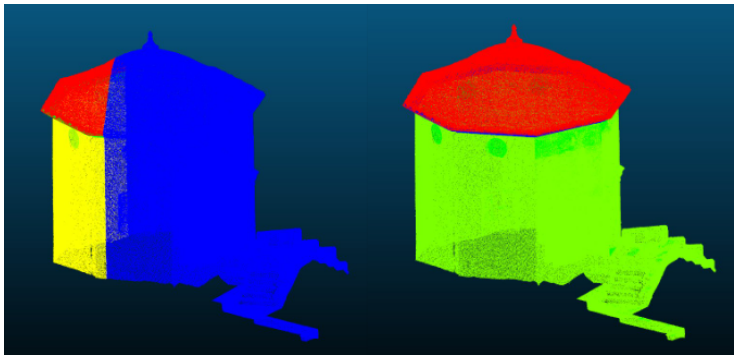

Figure 3.4: Example with Chapel

As we can see in Figure 3.4, the stairs and the door cannot be segmented because, in the initial segmentation, there are no points of those elements.

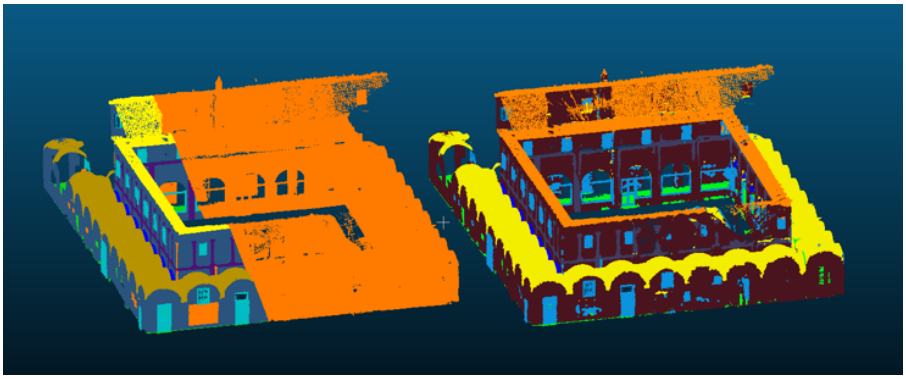

Figure 3.5: Example with Novalesa Cloister

In Figure 3.5, the Novalesa Cloister point cloud is segmented using this approach. The initial segmentation includes only a small portion of the point cloud, but it contains all

the classes present in the data. The classifier is able to correctly segment the entire point cloud, demonstrating the effectiveness of this method when the initial labeled subset is representative of all classes. However, some noise is still present, especially in some hard to segment classes.

3.1.7 Neural Network model approach

The approach

In this solution, the objective is to train a neural network model on the ArCH dataset and use it to segment new point clouds. Two model architectures were found suitable for this task: PointNet++ and DGCNN and they were both implemented with some modifications as explained earlier.

Following [10], the `Other` class is excluded from model training, since it is not a relevant class for the segmentation task and its heterogeneous content could confuse the neural network.

Class imbalance

A significant problem of the dataset used is the class imbalance.

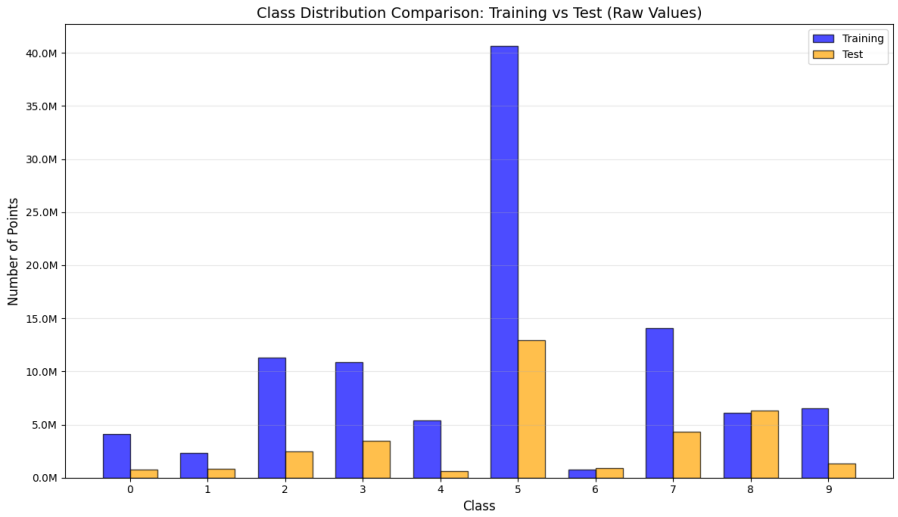

Figure 3.6: Class distribution

As shown in the Figure 3.6, the `wall` class is the most represented and `stairs`, `column`, `arch` are the least represented ones.

An high class imbalance can lead to a model that is biased towards the most frequent classes, resulting in poor performance on the less represented ones.

To address this issue some techniques were applied. The weights of classes are computed to handle them effectively during data preprocessing and model training.

Three weighting strategies were analyzed (Figure 3.7):

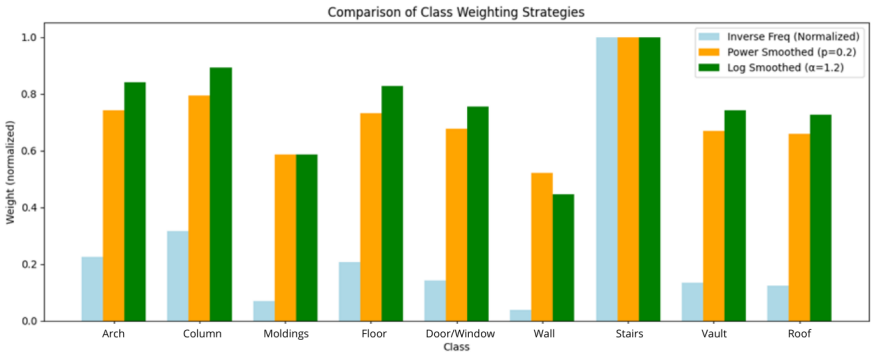

Figure 3.7: Class weighting techniques - Bar plot

- `Inverse frequency`: Each class weight is inversely proportional to its frequency:

$$w_i = \frac{N}{K \cdot n_i},$$

where N is the total number of samples, K the number of classes, and n_i the count of class i. This ensures that the sum of weights across classes is normalized, but may excessively penalize rare classes.

- `Power smoothed`: To reduce the sharp contrast between frequent and rare classes, the inverse-frequency weights are first normalized by their maximum, and then smoothed with an exponent $p \in (0,1)$:

$$w_i = \left(\frac{w_i^{(\text{inv})}}{\max_j w_j^{(\text{inv})}} \right)^p.$$

Smaller values of p shrink the gap between rare and frequent classes preventing instability while keeping relative differences.

- `Log smoothed`: Another approach (inspired from [7]) is to work with normalized frequencies $f_i = \frac{n_i}{N}$ and compute

$$w_i = \frac{1}{\log(\alpha + f_i)},$$

where $\alpha > 1$ is a smoothing constant. Weights are then normalized (e.g., dividing by $\max_j w_j$) for stability. This method mitigates the effect of very small frequencies and avoids excessively large weights for rare classes.

After this analysis, the log-smoothed technique was chosen and used as it is the most balanced approach.

Data pre-processing

For this approach, the data pre-processing involves several additional steps.

First, all the point clouds in the dataset are voxelized, and the centers of the voxels are taken as points.

All the point clouds present the same structure:

x y z r g b nx ny nz label

where:

- `x y z` are the point coordinates
- `r g b` are the color channels
- `nx ny nz` are the normals
- `label` is the class assigned to the point

Colors are converted into HSV format, as explained earlier, and a new column called `height` is added.

$$\text{height}_{\text{norm}} = \frac{z - \min(z)}{\max(z) - \min(z)}$$

This feature is a duplicate of the `z` column normalized in the whole point cloud; this column will remain unchanged in the next processing steps.

The final set of features is then:

x y z r g b nx ny nz height

Since neural network models require a fixed data size as input, all point clouds are split into 'cubes' of points (Figure 3.8).

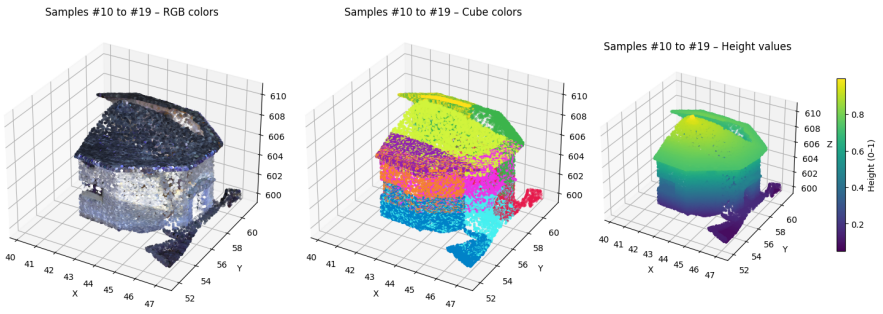

Figure 3.8: Example of cubes splitted with height values (overlap = 0.10, cube_size = 5)

To achieve this, a cube-shaped window is systematically moved throughout the point cloud. At each position, all points contained within the window are extracted and grouped together to form a single sample in the modified dataset.

This process is repeated by incrementally shifting the window across the entire spatial extent of the point cloud.

In this way, the point cloud is partitioned into multiple cubes, ensuring that every point is included in at least one sample. This strategy enables efficient handling of large point clouds by breaking them into manageable, fixed-size subsets suitable for neural network training and inference.

In order not to penalize the points on the edges, the window is shifted with a small overlap relative to its previous position. The shifting offset is computed as

$$\text{offset} = \text{cube\_size} - (\text{cube\_size} \times \text{overlap})$$

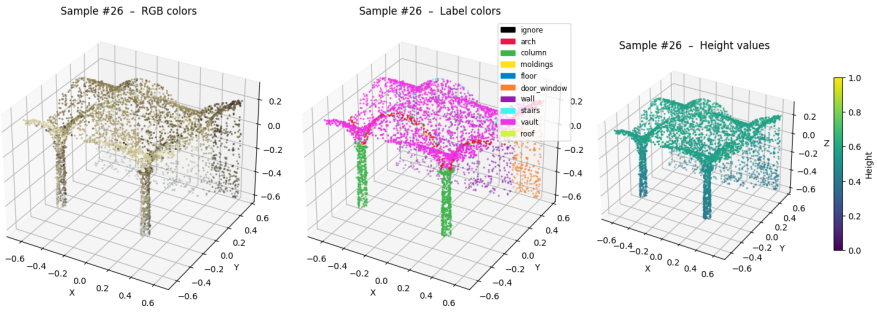

Figure 3.9: Example of a point cloud sample cube

For consistency, each sample is constrained to 4096 points.

To ensure that each sample fed to the neural network has a consistent size, every cube is processed as follows:

- **Downsampling:** If a cube contains more than the required number of points (4096), it is downsampled. Points are selectively removed, prioritizing the reduction of points from majority classes first (based on computed class weights), to avoid further penalizing underrepresented classes. This helps maintain a balanced class distribution within each sample and prevents the loss of rare class information.

- **Padding:** If a cube contains fewer points than required, padding is applied to reach the fixed size.

This approach ensures that all input samples have a uniform size, which is essential for efficient neural network training and inference, while also maintaining class balance and geometric consistency.

Padding points (with label `-1`) will be ignored during training. Two padding techniques were analyzed:

- **Interpolate padding**: This technique (Figure 3.10) adds new points between existing ones in a point cloud by interpolating their positions. It helps to create a denser point cloud, filling gaps and providing smoother representations of surfaces.

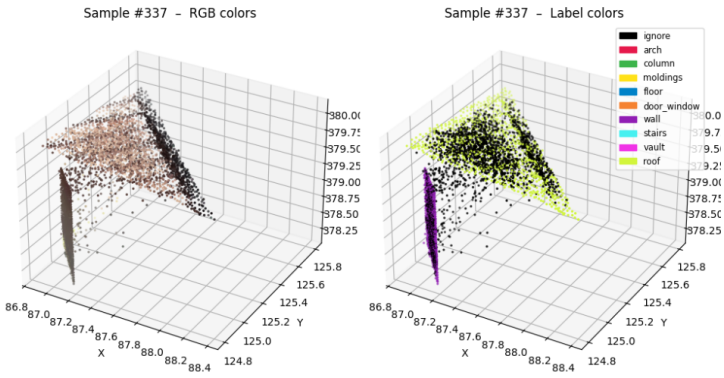

Figure 3.10: Example of interpolate padding

- **Adaptive sampling**: This approach (Figure 3.11) generates additional points by duplicating existing points, then applying small offsets to them to avoid exact overlap. This allows regions with insufficient points to reach a desired number of points while preserving the overall structure of the point cloud.

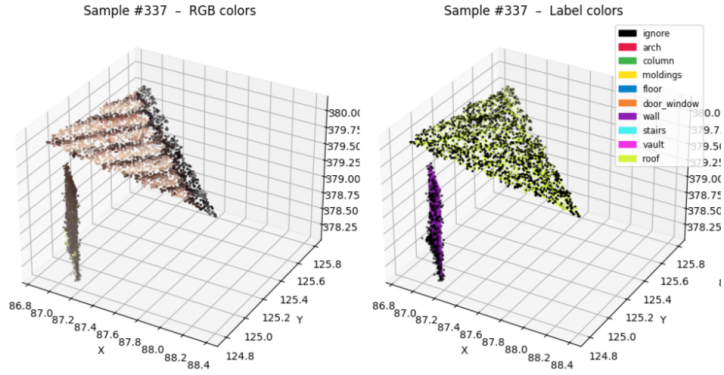

Figure 3.11: Example of adaptive sampling

Interpolate padding is generally more effective for PointNet++ as it enhances local surface density, while adaptive sampling better suits DGCNN by preserving graph connectivity without introducing artificial structures.

For this reason, both techniques were implemented in the final program and can be selected based on the chosen model architecture.

Data augmentation

The augmentation strategy was designed to preserve the geometric and semantic integrity of the architectural point clouds. Excessive transformations such as large random rotations, scaling, or aggressive noise injection can distort the spatial relationships and fine details that are critical for accurate segmentation of heritage structures.

Such distortions may result in samples that do not reflect real-world scenarios, causing the model to learn features that are not representative of the target domain.

To mitigate these risks, the following minimal augmentation techniques were employed:

- **Random rotation (z-axis only):** Rotations were restricted to the vertical axis (following [7]), which is consistent with the typical orientation of architectural scans and avoids unrealistic tilting of structures.
- **Small random jitter:** A minor random jitter was applied to point coordinates to simulate measurement noise and improve model robustness without altering the overall geometry.
- **Color jitter (HSV):** Slight random variations were introduced to the HSV color channels to account for lighting changes and sensor variability, while maintaining the original appearance of materials.

This conservative approach to augmentation ensures that the training data remains realistic and relevant, helping the model generalize to new point clouds while reducing the risk of overfitting.

Training

Model training, performance evaluation, and dataset analysis were conducted on Google Colab Pro and Kaggle platforms using Jupyter notebooks. In the next sections, the choices made for the training procedure are presented.

1. Loss Function A version of the Jaccard loss (also known as Intersection-over-Union, IoU, loss) optimized for GPU efficiency using the `torch` library is implemented and used during training.

Given model predictions of shape $[B, C, N]$ (batch, classes, points), the raw logits are first converted into probabilities by a softmax function along the class dimension. The ground-truth labels are then transformed into one-hot encodings of the same shape. A variable called `ignore_index` allows the masking of an irrelevant label, such as the padding class.

The IoU score is obtained as

$$\mathrm{IoU} = \frac{|A \cap B| + \varepsilon}{|A \cup B| + \varepsilon},$$

where ε is a small constant for numerical stability. The final loss is given by

$$\mathcal{L}_{\mathrm{Jaccard}} = 1 - \frac{1}{C} \sum_{c=1}^{C} \mathrm{IoU}_c.$$

The Jaccard loss is particularly powerful under class imbalance because it directly optimizes the IoU metric which focuses on the overlap between predicted and true regions rather than raw pixel/point counts. This reduces the bias toward majority classes and ensures that the minority classes contribute meaningfully to the loss.

2. Optimizer AdamW is a variant of the Adam optimizer that decouples weight decay from the gradient-based update.

While standard Adam incorporates weight decay by adding an ℓ_2 penalty term to the gradients, AdamW applies weight decay directly to the parameter update.

This adjustment decouples weight decay from the adaptive learning rates, resulting in more stable training and improved generalization.

In summary, the key difference is that Adam applies weight decay through gradient regularization, whereas AdamW decouples it as a separate step.

AdamW, by decoupling weight decay from gradient updates, helps prevent the majority classes from dominating the optimization process and encourages better generalization to minority classes.

3. Scheduler The ReduceLROnPlateau learning rate scheduler lowers the learning rate when a monitored validation metric (e.g., loss) stops improving for a specified number of epochs.

Unlike step-based schedulers, which decrease the learning rate after fixed intervals, ReduceLROnPlateau adapts dynamically to the training process, ensuring that the learning rate is reduced only when the optimization appears to have plateaued.

This allows the model to converge more effectively while avoiding unnecessarily small learning rates during periods of steady improvement.

ReduceLROnPlateau adaptively lowers the learning rate when progress stalls, allowing the optimizer to fine-tune decision boundaries for underrepresented classes instead of overshooting due to a high learning rate.

4. Metrics During training, performance is monitored using IoU, F1-score and accuracy.

IoU is a more reliable metric than accuracy in the context of class imbalance.

While accuracy can be dominated by majority classes (e.g., background points), IoU explicitly measures the overlap between predicted and ground-truth regions, penalizing both false positives and false negatives.

This makes IoU more sensitive to minority classes, providing a fairer evaluation of segmentation performance in point clouds.

The F1-score is also useful in the context of class imbalance because it balances precision and recall, capturing how well the model identifies minority classes without being biased by the majority class.

While IoU measures spatial overlap, F1 provides complementary insight into the trade-off between false positives and false negatives.

5. Models Modified versions of PointNet++ and DGCNN were implemented and trained.

Both introduce targeted enhancements to improve point cloud segmentation, particularly for complex and irregular structures such as heritage buildings.

Modified PointNet++: modifies the original architecture by explicitly including XYZ coordinates in each Set Abstraction (SA) layer while counting only non-coordinate

features.

Multi-Scale Grouping (MSG) is employed in the first two SA layers, while a single-scale approach is adopted in the third to avoid issues with coordinate handling.

A Transformer layer is added after the last SA layer to capture richer contextual information. Feature Propagation layers incorporate residual connections to propagate features better.

The segmentation head is enhanced with a residual block structure consisting of two convolutional layers separated by Batch Normalization, Leaky ReLU activation, and dropout. This design improves feature learning and stability.

Modified DGCNN: builds on standard DGCNN by replacing EdgeConv layers with residual EdgeConv blocks, preserving features across layers.

Local features from each EdgeConv block are concatenated with a global feature vector to provide context for every point.

The segmentation head is deeper, using sequential Conv-GroupNorm-LeakyReLU blocks with dropout, which enables more robust learning for fine-grained segmentation.

Overall, these modifications in both networks enhance their ability to capture detailed local structures and global shape information, resulting in improved performance.

6. Training with Curriculum Learning To enhance model performance, the training procedure is divided into two distinct stages.

The first stage employs a Curriculum Learning strategy, where the model is trained on a reduced subset of the data that contains mostly the samples with underrepresented or more challenging classes.

In this case, the chosen classes are *arch*, *column*, *door_window*, and *stairs*.

Emphasizing these classes helps the model learn their features more effectively before training on the full dataset.

This stage uses a slightly higher learning rate and runs for a few epochs, allowing the model to capture essential patterns for these difficult classes quickly.

In the second stage, training continues on the full dataset with a lower learning rate, enabling the model to refine its representations and achieve more stable convergence across all classes.

Inference

To perform inference on a new point cloud, the data must be pre-processed in the same way as done to the dataset during model training.

So, when a new point cloud is loaded, it is voxelized, the colors are converted into HSV format, the normals are computed if not present, and the point cloud is split into cubes of points.

It is essential to track the points transformations to reconstruct the original point cloud with the corresponding segmentation labels.

Specifically, at each point is assigned a unique index and voxelization is performed while preserving the indices of the points contained within each voxel.

After segmenting the voxelized sample, the original points are restored by assigning them the label of the corresponding voxel center.

This procedure is applied both to the full point cloud initially and during cube splitting when downsampling is required.

Additionally, HSV colors must be converted back to RGB format after the segmentation.

3.1.8 Results

The following results belong to the main training phase, which was conducted after the initial curriculum learning stage. Metrics and analyses presented here reflect the model's performance during full training on the entire dataset.

ModPointNet++

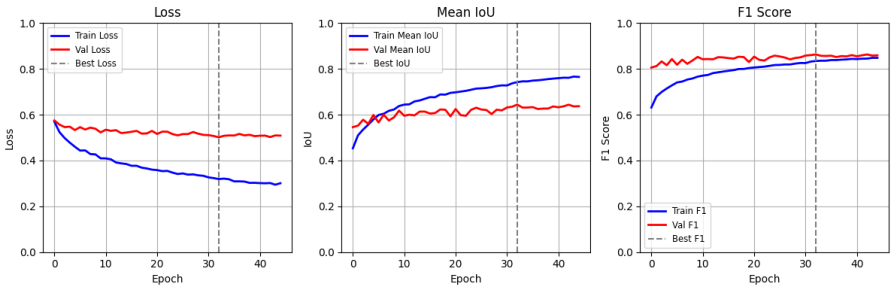

Figure 3.12: Metrics across epochs

In the Figure 3.12, the training and validation metrics for the modified PointNet++ model are presented.

Both training and validation losses show a gradual decrease over time, which indicates that the model is learning from the data. Importantly, the validation loss does not increase sharply compared to the training loss, suggesting that the model is not strongly overfitting even though the dataset is imbalanced.

The Intersection over Union (IoU) and F1 scores for the validation set also improve steadily, even though more slowly than those for the training set. This pattern shows that the model is generalizing well and learning meaningful features, rather than simply memorizing the training data.

Overall, the metrics remain at moderate levels. This outcome reflects the inherent complexity of the dataset and the relatively small size of the validation set, which can make it harder for the model to achieve very high scores.

A high F1 score (above 0.8) indicates that the model is effectively handling class imbalance and is able to correctly identify most classes. However, the IoU metric, which is more sensitive to precise segmentation boundaries, remains lower.

The analysis of the training and validation per-class IoU curves (Figure 3.13) shows that the model performs unevenly across different classes.

During training, all classes show a steady improvement, with some reaching values above 0.8, while others remain consistently lower, converging around 0.6. In validation,

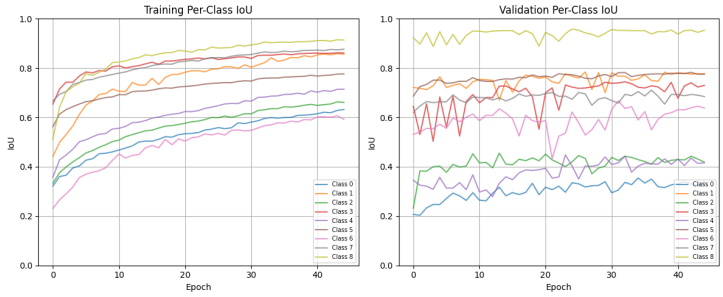

Figure 3.13: Per-class IoU

the same classes achieve high IoU values, while the weaker ones continue to be problematic. Furthermore, the validation curves reveal significant fluctuations for certain classes, indicating instability likely caused by class imbalance, limited validation samples, and high intra-class variability.

There is no evidence of severe overfitting considering that the training and validation behaviors remain broadly aligned.

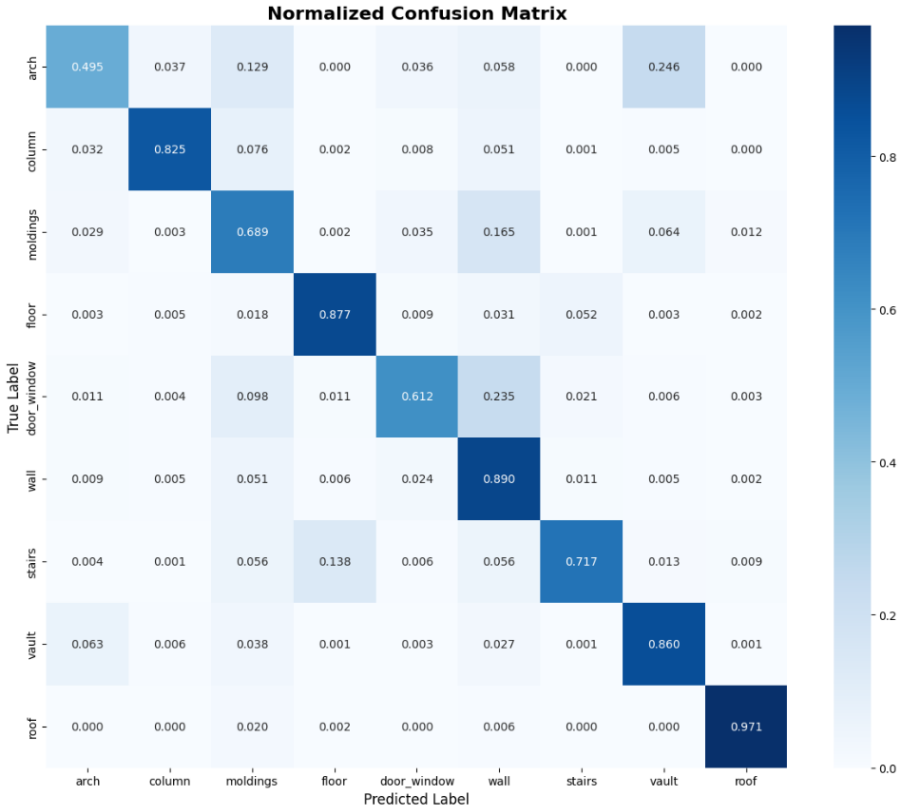

Figure 3.14: Confusion Matrix Normalized

The confusion matrix shown in Figure 3.14 is normalized to make it clearer where the

model is making mispredictions.

As we can see, the model is predicting 25% of `arch` points as `vault` and 13% as `moldings`, 24% of `door_window` points as `wall` and 14% of `stairs` points as `floor`.

As explained in [10], the low performance values are mainly due to geometric similarities between certain elements, the heterogeneous nature of some classes, and limitations in the data acquisition process.

In particular, some categories share very close geometric features leading to confusion between them and other categories group highly diverse objects making them harder to classify consistently.

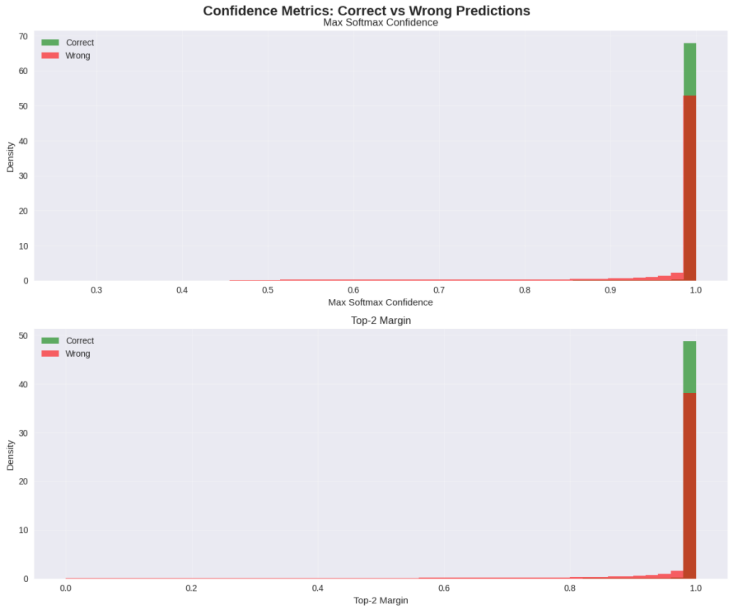

Figure 3.15: Confidence and Top-2 Margin

In Figure 3.15 further analysis on model predictions are presented.

The *confidence* (max softmax confidence) corresponds to the probability assigned to the most likely class. A higher confidence indicates that the model is more certain about its prediction. The *top-2 margin* is defined as the difference between the highest and the second-highest predicted probabilities. A large margin reflects that the model strongly favors one class over the others, whereas a small margin indicates that the prediction is more uncertain or ambiguous.

In this case, the model demonstrated high confidence for the correct predictions, while the misclassifications were generally associated with more uncertainty.

In Figure 3.16 the 10 most confused pairs of classes are analysed. The uncertain samples are computed by selecting those where the probability margin between the top-1 and top-2 predictions falls below a predefined threshold (0.5).

For these uncertain predictions, we count the occurrences of each class pair, sorting the pairs by frequency to determine the top 10 most confused pairs.

This analysis highlights which class pairs the model struggles to distinguish, revealing

Figure 3.16: Confused Pairs

specific weaknesses.

The results are aligned with the confusion matrix with a peak confusion between `moldings` and `wall` also due to the high number of points in the `wall` class.

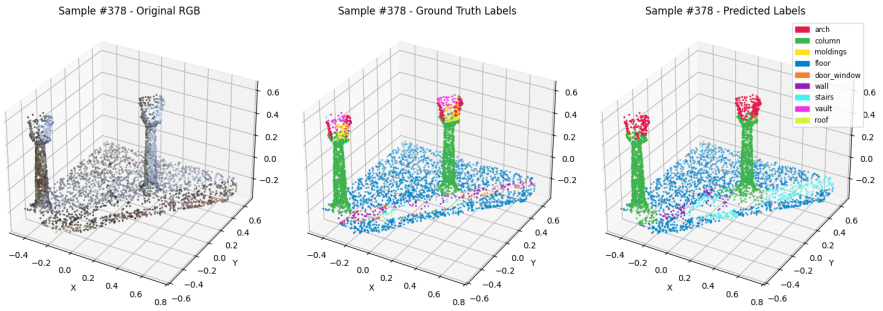

Figure 3.17: Example of a cube prediction (ModPointNet++)

ModDGCNN

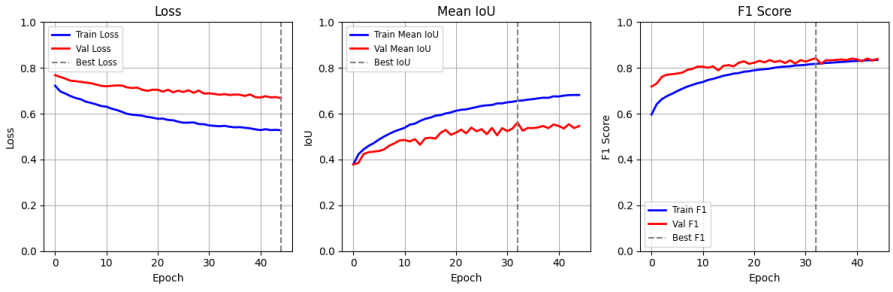

Figure 3.18: Metrics across epochs

The DGCNN model training metrics (Figure 3.18) show progress similar to the PointNet++ ones but with slightly lower performance.

Following the initial training phase, as no clear signs of overfitting were observed and the performance continued to show slight improvements, the process was extended for additional epochs to refine the model further.

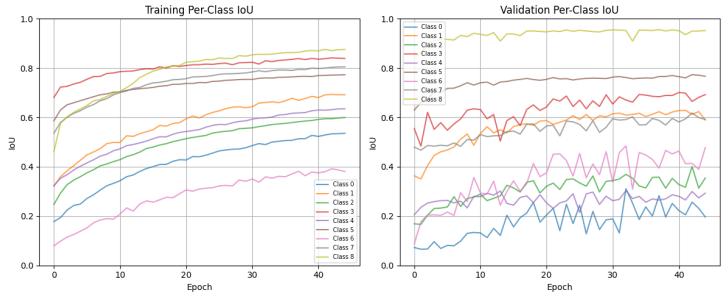

Figure 3.19: Per-class IoU

The per-class IoU performance (Figure 3.19) in this case turned out to be slightly lower compared to that of the PointNet++ model. By examining both plots, we can see that the learning curves on the training set remain relatively smooth and stable, while those on the validation set exhibit significant fluctuations. This instability is particularly evident in classes that are underrepresented in the dataset (especially in the validation set), since even small variations can heavily impact the IoU metric for those categories.

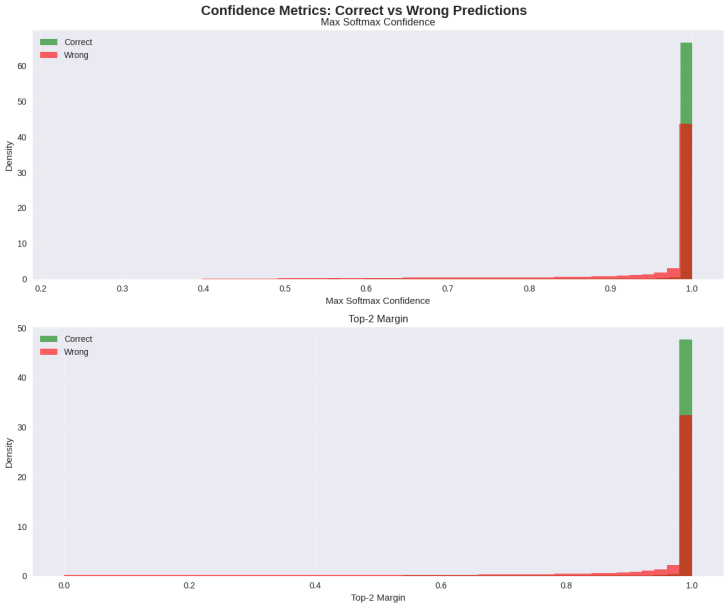

Figure 3.20: Confidence and Top-2 Margin

The confidence plot and the top-2 margin plot (Figure 3.20), compared to the ones of PointNet++, does not show major differences overall; however, the model appears less confident in its predictions, especially for those that are incorrect.

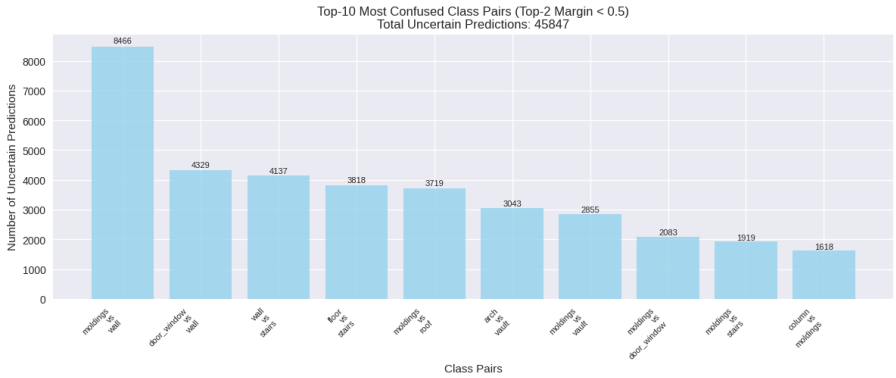

Figure 3.21: Confused Pairs

Both models share similar top confusions (moldings-wall, door_window-wall, floor-stairs, moldings-roof), but the absolute numbers are consistently higher in DGCNN (Figure 3.21), indicating more uncertainty.

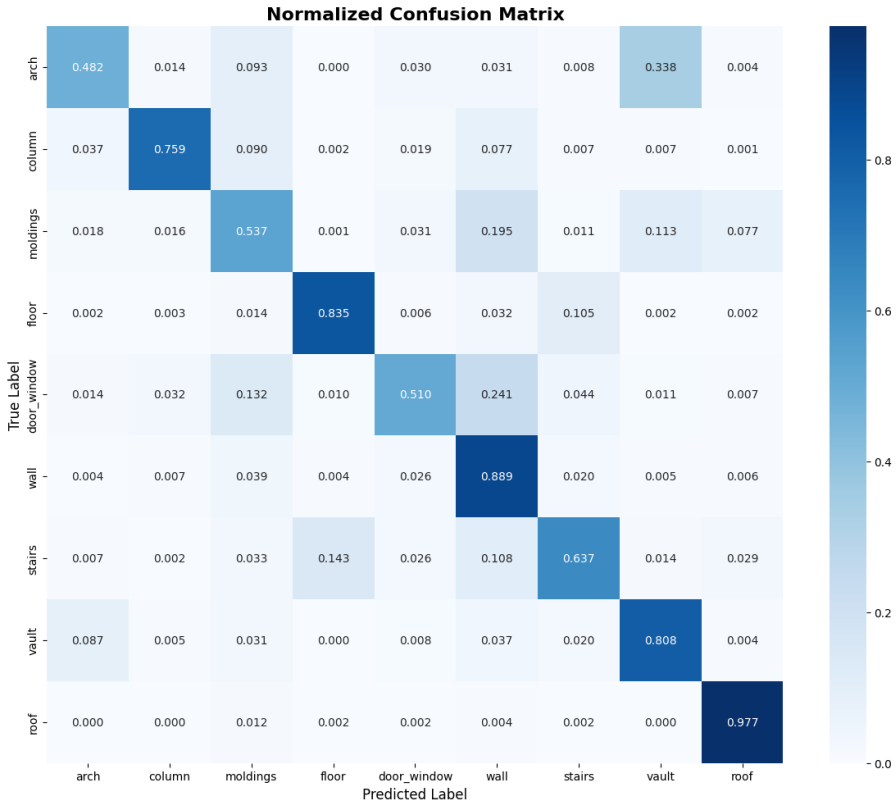

Figure 3.22: Confusion Matrix Normalized

The confusion matrix (Figure 3.22) shows similar results to those of the PointNet++ one, with slightly worse results.

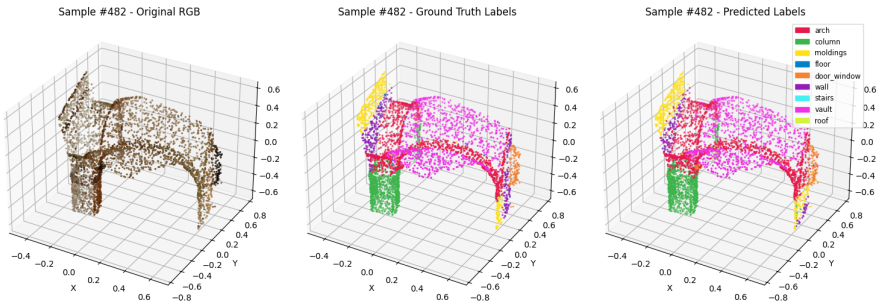

Figure 3.23: Example of a cube prediction (DGCNN)

Final models results

In the following section, the final evaluation metrics of the models are presented, followed by a qualitative analysis of their segmentation results on representative point clouds.

These qualitative results help to contextualize the numerical metrics demonstrating how the models perform on real-world data and where they may struggle due to data quality, class imbalance or architectural limitations.

Overall, this comprehensive evaluation combines both quantitative and qualitative perspectives to provide a clear understanding of the models' capabilities and areas for future improvement.

Model	**Loss**	**IoU**	**F1 Score**
PointNet++	52.84	60.35	85.69
ModPointNet++	50.14	65.05	88.03
DGCNN	53.84	56.92	83.74
ModDGCNN	62.21	58.80	85.35

Table 3.1: Comparison of evaluation metrics (%)

For reference, basic versions of the models are also included to illustrate the improvements achieved through the model modifications. Table 3.1 presents a comparison of the metrics for the final trained models.

The modified versions of the models show improved performance compared to their basic counterparts with ModPointNet++ achieving the highest IoU and F1 scores.

Table 3.2 presents the per-class IoU metrics for each model. Overall, the modified version of PointNet++ achieved the strongest performance among the tested models.

Compared to the basic PointNet++ and DGCNN architectures, ModPointNet++ consistently yields higher IoU scores across most classes, especially for challenging categories such as `column`, `moldings`, and `door_window`.

For instance, ModPointNet++ improves the IoU for `door_window` from 25.27% (PointNet++) to 45.33% and for `column` from 68.95% to 76.83%. The DGCNN models, while competitive for certain classes like `wall` and `roof`, generally fall behind in overall and per class IoU especially for underrepresented classes such as `stairs` and `arch`.

Model	arch	column	moldings	floor	door/window
PointNet++	36.61	68.95	37.51	74.79	25.27
ModPointNet++	33.72	76.83	44.64	75.54	45.33
DGCNN	31.13	67.78	34.96	62.54	37.54
ModDGCNN	33.20	67.70	37.41	72.36	35.54

(a) Classes arch-door/window

Model	wall	stairs	vault	roof
PointNet++	75.92	61.42	67.38	95.28
ModPointNet++	79.35	63.93	69.61	96.54
DGCNN	78.26	48.66	60.54	90.87
ModDGCNN	77.83	49.64	59.89	95.61

(b) Classes wall-roof

Table 3.2: Per-class IoU comparison (%)

These results highlight that the architectural modifications such as enhanced feature propagation and contextual layers lead to more balanced and robust segmentation, especially for minority and geometrically complex classes.

Predicted point clouds

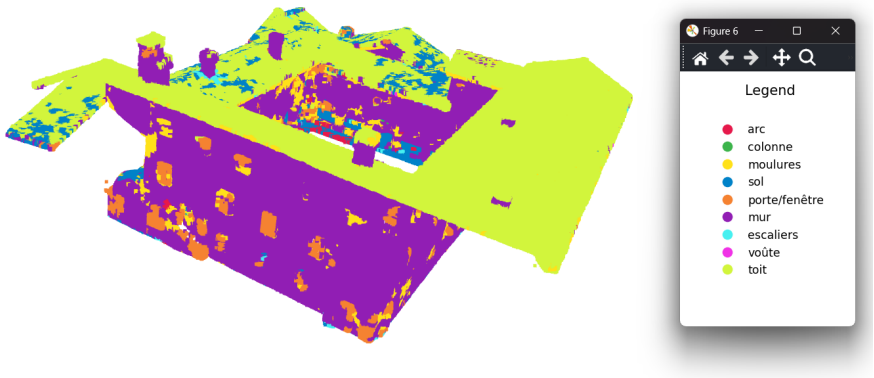

Figure 3.24: Novalesa Cloister segmented with ModPointNet++

Although PointNet++ outperforms DGCNN in overall quantitative metrics, visual assessment indicates that DGCNN (Figure 3.25) produces more accurate segmentation of `columns` and `doors_windows` than PointNet++ (Figure 3.24) in this point cloud example.

However, DGCNN's results exhibit higher variability whereas PointNet++ produces more consistent segmentations.

This outcome is specific to this point cloud, and results may vary with different data.

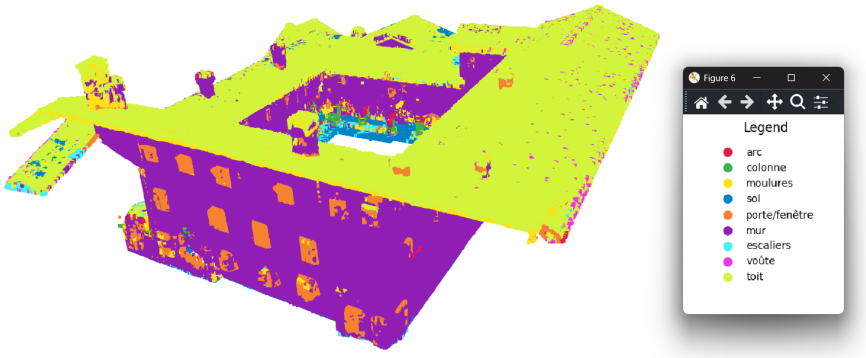

Figure 3.25: Novalesa Cloister segmented with ModDGCNN

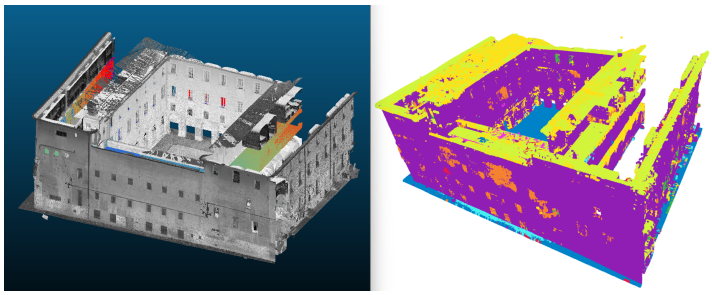

Figure 3.26: Lycée Sainte Catherine segmented with ModDGCNN

In Figure 3.26, the Lycée Sainte Catherine point cloud is segmented using ModDGCNN. We can notice that, since colors were incorporated during model training, the absence of color information in a point cloud can lead to worse results.

3.1.9 Discussion

The results indicate that the models learned to recognize certain patterns within the point cloud; however, segmentation lacks precision.

This limitation is attributed in part to the variability in data quality and also to data scarcity for certain classes.

In Figure 3.27 we can see a detail of a point cloud from the dataset that displays significant quality issues.

Some areas are overexposed with very bright colors and the point sampling contains a significant amount of noise, which badly affects the segmentation performance. In the dataset used, also other point clouds present similar quality issues.

After applying pre-processing and analyzing different model architectures, the modified PointNet++ shows a slight performance improvement.

However, the results remain insufficient for reliable predictions on new point clouds, as certain classes are still poorly segmented but this work provides a solid foundation for further development.

Figure 3.27: Detail of Novalesa Cloister point cloud

3.1.10 Next Steps

To further strengthen the models, the most important requirement is a richer and more diverse annotated dataset.

A larger dataset would allow the model to learn more robust representations of the data and to generalize more effectively.

Another promising direction is the incorporation of geometric consistency rules to correct potential misclassifications.

For example:

- The points of the doors and windows should always be surrounded by the points of the wall.
- The wall points should always be located above the floor points.
- Stair points should not appear among the roof points.

Enforcing such constraints after the prediction can significantly improve output quality.

This approach can be extended by grouping segmented points into object-level entities and constructing a relational graph of their spatial configurations.

Using these relationships, inconsistencies in segmentation can be detected and corrected, further improving the reliability of the model.

Additionally, exploring ensemble methods that combine predictions from multiple models could enhance robustness and accuracy.

3.1.11 Final Program

A standalone program was developed to allow users to load and segment new point clouds using the trained models. The program interface is in French and accepts point clouds

in `txt`, `las`, and `e57` formats as input and the segmented point clouds are saved in `las` format.

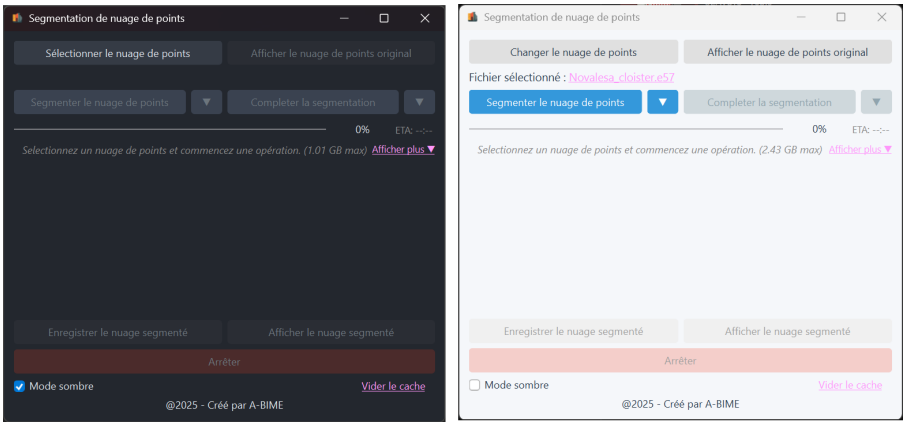

Figure 3.28: Standard program interface (dark and light mode)

The final program, as shown in Figure 3.28, has a simple interface to load and segment new point clouds.

Theme mode (dark or light) is automatically matched to the current system one and can be toggled through the button in the bottom left corner.

A progress bar with ETA computation and a small description is present to show the current operation progress. Also, a description box can be toggled to show/hide additional information about the current and previous operations.

Neural Network segmentation parameters

Figure 3.29 shows the available parameters of the approach using the trained NN models:

- *Model:* allows the user to select the segmentation architecture, either `PointNet++` or `DGCNN`. Each model has different strengths and may perform differently depending on the chosen point cloud.
- *Voxel size:* defines the resolution of the point cloud during the voxelization step. Smaller voxels preserve more geometric detail but significantly increase memory usage and computation time while larger voxels reduce the dataset size at the cost of fine details.
- *Cube size:* specifies the dimensions of the sub-cubes into which the point cloud is split before inference. Larger cubes speed up processing but risk losing local detail, whereas smaller cubes allow for finer segmentation but increase the total number of them.
- *Cube overlap:* controls the degree of overlap between adjacent cubes. A higher overlap increases inference time but generally improves segmentation accuracy, since overlapping regions are resolved using **majority voting**, reducing edge artifacts at cube boundaries.

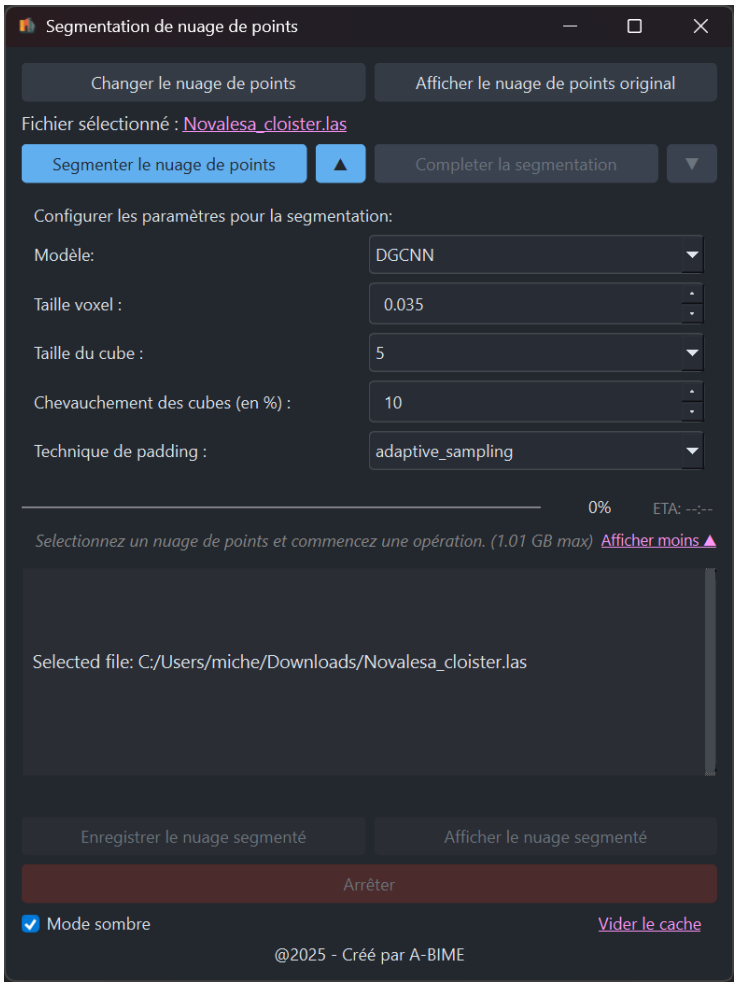

Figure 3.29: Neural Network segmentation parameters

- *Padding technique:* determines how to handle cubes with fewer points than required. Two strategies are available: **adaptive sampling**, which varies density but preserves geometry, and **interpolation**, which adds points interpolating the existing ones.

ML Classifier parameters

The button to complete the segmentation is automatically enabled/disabled depending on whether an initial classification is present in the loaded point cloud or not.

Figure 3.30 shows the available parameters of the approach using the ML classifier:

- *Class Index:* Applied only to `txt` files, this specifies the index of the column containing the point labels. The `auto` option detects it automatically.
- *Number of Neighbors (k):* The number of nearest neighbors considered for feature extraction of the points.

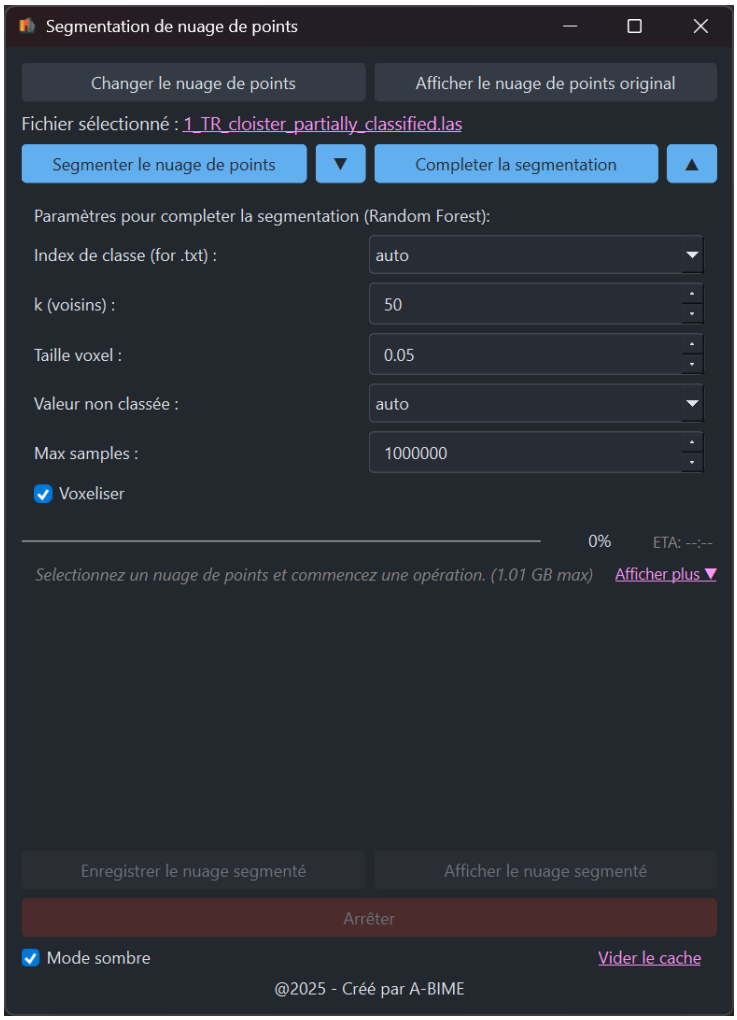

Figure 3.30: Segmentation Completion parameters

- *Voxel Size:* Defines the size of voxels used in the voxelization step, as previously described.

- *Not Classified Label Value:* The label assigned to points that have not yet been classified. By default is automatically detected, but it can be manually set to a specific number.

- *Max Samples:* The maximum number of points taken from the segmenteed part of the point cloud to train the classifier, useful for controlling memory usage and training time.

3.2 Manuscripts Handwritten Text Extraction

3.2.1 Task

The main objective of this project is to design a pipeline capable of extracting text from digitized images of old manuscripts.

Once the text is extracted, the system should allow large-scale analysis of historical documents by automatically searching specific keywords. For instance, given a corpus of 600 manuscript pages, the program should generate a structured output (e.g., a CSV or Excel file) in which each document is associated with the list of detected keywords.

To ensure robustness, the set of keywords should be predefined and enriched with multiple variations and conjugations.

This makes the system more reliable when dealing with linguistic differences, archaic spellings, or variations in writing style. Assistance from AI tools can be employed to expand the keyword list semantically and morphologically.

3.2.2 AI Approach

The initial approach explored involves leveraging generative AI models to perform the text extraction task.

Vertex AI is a fully managed and unified platform provided by Google Cloud for building and deploying AI solutions including generative AI and computer vision tasks.

Some tests were already done in the company with Vertex AI using a two-step prompting strategy:

- *Prompt 1:* Instruct the AI to analyze the manuscript image and extract the raw text, preserving the original formatting as much as possible.
- *Prompt 2:* Provide the extracted text to the AI and ask it to correct any errors, improve readability and format it according to specific guidelines.

While this approach provided an acceptable baseline, the AIs lack of specialization in historical text recognition led to frequent errors and inconsistencies.

Possible Improvements

Image Preprocessing: A possible improvement is to add a preprocessing step to enhance the manuscript images before text extraction. This could be achieved through an additional prompting step (*Prompt 0*) instructing the AI to:

- Remove stains and artifacts.
- Restore degraded or broken parts of the sheet.
- Increase the contrast between text and background (e.g., black text on white background).

By improving the visual quality of the input, subsequent text recognition becomes easier and more accurate.

If the model cannot directly output enhanced images, the preprocessing instructions could be incorporated directly within the first text-extraction prompt.

Guided Text Extraction: Another way to improve performance is to guide the AI in the extraction process. Instead of directly asking for text, the model can be instructed to:

- Identify and segment lines of text before transcribing them.
- Recognize structural elements such as titles, paragraphs, or marginal notes.
- Take into account document-specific metadata (e.g., type of manuscript, historical period, or language variant).

Providing such contextual and procedural guidance allows the AI to perform a more reliable and structured transcription.

Limits: Despite its flexibility, this approach has several limitations.

First, creating a fully automated pipeline is challenging, as the process relies on multiple sequential prompts and often requires manual intervention to correct errors or inconsistencies.

Second, since Vertex AI is a general-purpose AI and not specialized in historical text recognition, the extracted text may contain inaccuracies, misinterpretations, or formatting errors that reduce overall precision.

Lastly, the cost of using a paid service like Vertex AI can become significant, especially when processing large volumes of documents.

3.2.3 Existing ML Tools

Most used ML tools in this field are Transkribus [14] and eScriptorium [5].

Transkribus is an ML-based tool for historical text recognition, enabling automatic transcription, seamless editing, and collaborative work. It also allows users to train custom models tailored to specific document types.

eScriptorium provides similar ML-based functionalities for text recognition and custom model training.

The main difference lies in accessibility: while Transkribus is a licensed product available only via subscription, eScriptorium is free and open-source.

This allows full access to its underlying code, APIs, and model training routines, enabling developers to adapt and extend its functionality to specific workflows. Unlike Transkribus, which is a licensed and closed platform, eScriptorium can be deployed locally or on a private server, facilitating automation without relying on external services or subscriptions. This flexibility makes it possible to seamlessly combine image preprocessing, text extraction, and post-processing steps into a handmade pipeline tailored to the project's requirements, ensuring reproducibility, customization, and scalability.

With eScriptorium, it is possible to:

- Use pre-trained models for initial text extraction.
- Segment documents into lines or regions of text.
- Train custom models on specific datasets to improve accuracy.
- Manually correct and refine transcriptions.

Exploiting these features, it is possible to create a highly efficient and tailored workflow for historical document transcription and analysis adapted to the project's needs.

3.2.4 Custom DL/ML Approaches

Custom deep learning and machine learning approaches are the best option to achieve the highest accuracy in handwritten text recognition and the maximum integration flexibility, but they come with specific requirements:

- **Annotated dataset:** A rich dataset of already annotated images is essential. Each image must have a corresponding text file representing the ground truth. For training robust models, as suggested in an eScriptorium webinar, at least 25–75 pages (5,000–15,000 words) of annotated data should be available. This requirement applies not only to fully custom models but also to custom models in platforms like Transkribus and eScriptorium. Annotation can be performed manually or semi-automatically, where a pre-existing model is used for initial predictions and then corrected manually.
- **Hardware:** Training deep models requires significant computational resources. Platforms like Google Colab or Kaggle provide suitable environments for experimentation and training.
- **Data specificity:** A model trained on certain data types will perform well on similar data but may struggle with other styles or document types. Therefore, the dataset should be representative of the target documents.
- **Model architecture and preprocessing:** Careful selection of neural network architecture and preprocessing strategies is crucial for achieving good performance.

Several machine learning approaches have been developed to extract text from historical manuscripts. These methods typically combine convolutional neural networks (CNNs) for visual feature extraction with recurrent neural networks (RNNs) for sequence modeling.

Deep Convolutional and Recurrent Neural Networks

El Bahi [2] presents a hybrid deep learning system for Handwritten Text Recognition (HTR) and information extraction from ancient manuscripts, combining convolutional and recurrent neural networks in an end-to-end architecture. The approach consists of several key steps:

1. **Preprocessing:** Manuscript images go through normalization, grayscale conversion, deskewing, and artifact reduction to enhance quality and mitigate degradation effects typical of historical documents. Data augmentation is also applied to simulate handwriting variations and improve model generalization.

2. **Feature Extraction (CNN):** A convolutional neural network (CNN), inspired by VGG16, that automatically extracts robust spatial features from the input images. These layers capture local visual patterns and hierarchies in handwriting, effectively reducing raw images to meaningful feature maps even in the presence of noise.

3. **Sequence Modeling (RNN):** The extracted feature maps are fed into a recurrent neural network (RNN), typically using LSTM or GRU units, to model temporal dependencies and character sequences. This enables the system to understand the context and structure of handwritten lines, improving recognition accuracy by considering character order and co-occurrence.

4. **Prediction and Training:** The CNN and RNN components are integrated into a unified Convolutional Recurrent Neural Network (CRNN) architecture. Connectionist Temporal Classification (CTC) loss is used for training, allowing the model to learn without explicit alignment between image segments and text labels.

5. **Information Extraction:** Beyond text transcription, the system leverages the contextual understanding from the RNN layers to extract named entities and key information relevant to ancient texts.

Evaluation on historical manuscript datasets demonstrates the effectiveness of this CRNN approach in handling diverse handwriting styles, complex document layouts, and limited annotated data. By combining visual feature extraction and sequence modeling in an end-to-end trainable system, this method achieves state-of-the-art results for offline handwritten text recognition and information extraction in challenging ancient manuscript images.

CRNNs with Limited Labeled Data

Other approaches employ deep Convolutional Recurrent Neural Networks (CRNNs) also inspired by VGG16 [4]. These systems stack multiple convolutional layers followed by bidirectional LSTM layers to capture both local and sequential patterns in text-line images. Spatial pooling, ReLU activation, and a sliding window approach are used to enhance feature extraction and sequence prediction.

The model consists of 13 convolutional layers (3x3 filters, stride 1x1) for feature extraction, followed by three bidirectional LSTM layers (256 units each) to model contextual dependencies in character sequences. Input text-line images are normalized and scanned with a sliding window (64x64 pixels), extracting feature vectors from the last convolutional layer to feed the recurrent layers.

Training is performed end-to-end using Connectionist Temporal Classification (CTC) loss, which enables learning without explicit alignment between input and output sequences as a key advantage for handwriting recognition tasks. To address the challenge of limited labeled data, the authors employ incremental training: initially, only 10% of text-line data is manually labeled to train a bootstrap model. This model is then used to recognize the remaining lines and the most confidently recognized lines (validated via Levenshtein distance) are added to the training set for further refinement.

Multi-scale data augmentation is applied by synthetically scaling text-line images into three size classes (Small, Medium, Large), improving robustness to handwriting scale variability. During inference, a model-based normalization scheme is used: test images are normalized at multiple vertical scales, and recognition scores are combined using the ROVER method to further enhance accuracy.

Optimization is performed using the Adam optimizer, with batch normalization after convolution layers and ReLU activation for improved convergence and generalization. This approach achieved the second-best result in the ICDAR 2017 competition on the READ dataset, demonstrating efficient use of limited ground truth, robust handling of scale variability, and a practical workflow for high-accuracy offline handwritten text recognition in historical documents.

Region-Based Document Segmentation with dhSegment

DhSegment [1] is a deep learning model specifically designed for document layout analysis and segmentation, rather than direct handwriting recognition. DhSegment employs a fully convolutional network (FCN) architecture with a ResNet backbone to extract semantic features at multiple scales from historical document images.

The model performs pixel-level classification generating segmentation masks for various layout components such as text regions, illustrations, decorations, and tables. Its architecture combines convolutional layers, skip connections, and up-sampling (decoder) layers, enabling the production of detailed and spatially accurate masks.

A key advantage of dhSegment is its adaptability: it can be trained on relatively small annotated datasets and easily repurposed for different segmentation tasks, including page extraction, text line detection, and zone segmentation in manuscripts. This flexibility makes it suitable for handling the diverse and complex layouts typical of historical documents.

While dhSegment does not perform handwriting recognition itself, its segmentation output is a crucial preprocessing step. By isolating text regions from noisy or heterogeneous backgrounds, it enables downstream handwriting recognition models (such as CRNNs) to operate on cleaner, well-defined inputs, thereby improving transcription accuracy.

In summary, dhSegment provides a generic, end-to-end deep learning solution for

document segmentation, leveraging a ResNet backbone and FCN-style pixel-wise classification. Its ability to efficiently learn from limited samples and adapt to various segmentation challenges makes it a valuable component in workflows for historical manuscript analysis, especially as a precursor to handwriting recognition.

3.2.5 Datasets

In the following section, a selection of manuscript datasets suitable for model training is presented. As noted earlier, it is important to choose a dataset that closely aligns with the style and structure of the target documents.

Using a dataset that is too different from the target documents may lead to poor model performance due to discrepancies in handwriting styles, languages, or document layouts.

If custom datasets are available, open source datasets can still be used to augment the training data and improve model robustness.

The ESPOSALLES Database

The ESPOSALLES Database [6] is a large-scale historical handwriting and data-mining project by the Document Analysis Group (DAG) at the Computer Vision Centre (CVC), part of UAB in Barcelona. It consists of:

- A digitized archive of marriage registers known as *Llibres d'Esposalles*, preserved at the Archives of Barcelona Cathedral.
- 291 books documenting roughly 600,000 marriages from 250 parishes, spanning the years 1451 to 1905.
- Full handwritten transcription at line and word level, creating structured, searchable text from manuscript images.
- Designed for document analysis tasks, including handwriting recognition, indexing, browsing, and image/text querying.

The ESPOSALLES Database is one of Europe's most extensive annotated historical datasets for handwritten marriage records, making it ideal for research in palaeography, social history, genealogy, and machine learning on historical documents.

CATMuS/Medieval

The Consistent Approaches to Transcribing Manuscripts (CATMuS [3]) dataset offers:

- A framework for annotation practices of medieval manuscripts.
- A benchmarking setup for assessing automatic text recognition models with respect to diverse variables such as historical period, linguistic variety, textual genre, and writing system.

- Support for other tasks such as script classification, dating, and line-based computer vision tasks.

CATMuS, developed in collaboration with several institutions, covers a corpus of over 200 manuscripts written in 10 languages, from the 8th to the 16th century. With more than 160,000 transcribed lines and 5 million characters, and adhering to consistent transcription principles, it serves as a robust benchmark for the evaluation of Handwritten Text Recognition (HTR) models applied to historical material.

The dataset includes a wide range of documents in different languages spanning Middle and Old French, Middle Dutch, Catalan, Spanish, Navarese, Italian, Venetian, Old English, and Latin. All materials are made available under a CC-BY license, ensuring open access and reusability.

ICDAR2017 HTR Dataset (READ Dataset)

The ICDAR2017 Handwritten Text Recognition competition dataset [13] consists of several subsets:

- **Train-A:** Pages with manually revised baselines and corresponding transcripts (50 pages). Only baselines are corrected; line polygons are not reviewed.
- **Train-B:** Pages without layout or line information; transcripts are provided at page level (10k pages, divided into two 5k-page batches).
- **Test-A:** Pages with manually revised baselines (65 pages). Line polygons not reviewed.
- **Test-B1:** Test-A pages annotated with only region geometry; text line information not provided.
- **Test-B2:** Page images annotated with regions for detecting and recognizing text lines (57 pages).
- **Baseline.tgz:** Baseline system trained using the first 40 pages of Train-A, based on the Laia deep learning toolkit for handwritten text transcription.

3.2.6 Discussion

In conclusion, while generic neural network architectures provide strong baselines for segmentation and recognition tasks, the specific challenges of text extraction in manuscripts (such as irregular layouts, varying handwriting styles, degradation of the support, and the presence of non-textual elements) make them less efficient without extensive adaptation.

A custom machine learning approach, on the other hand, allows the integration of domain-specific features, tailored preprocessing strategies and targeted classifiers that directly address these peculiarities.

This flexibility not only improves accuracy and robustness but also ensures better interpretability and adaptability to diverse manuscript collections. Consequently, a custom

ML pipeline emerges as a more reliable and effective solution for extracting text from historical manuscripts.

Once a robust ML pipeline is established, it can be expanded to automatically handle a large number of documents and generate structured outputs for further analysis, such as keyword searches or metadata extraction. A fully automated system would significantly enhance the efficiency of historical document analysis and conservation, enabling researchers to focus on interpretation and insights rather than manual transcription.

3.2.7 Future Directions

To further advance the field of handwritten text extraction from historical manuscripts, several promising research directions can be considered:

- **Transfer Learning and Domain Adaptation:** Leveraging pre-trained models on large, generic handwriting datasets and fine-tuning them on smaller, domain-specific collections can significantly improve recognition accuracy, especially when annotated data is scarce.
- **Self-Supervised and Semi-Supervised Learning:** Employing self-supervised or semi-supervised techniques can help utilize unlabeled manuscript images, reducing the reliance on extensive manual annotation and enabling models to learn robust representations from raw data.
- **Multimodal Approaches:** Integrating visual features with contextual metadata (such as document provenance, historical period, or linguistic information) can enhance model performance and facilitate more accurate extraction of structured information.
- **Active Learning:** Implementing active learning strategies, where the model identifies uncertain or ambiguous samples for manual review, can optimize annotation efforts and accelerate the creation of high-quality training datasets.
- **Explainable AI:** Developing interpretable models and visualization tools can help historians and archivists understand model decisions, identify sources of errors, and build trust in automated transcription systems.
- **End-to-End Pipelines:** Building fully automated, modular pipelines that combine preprocessing, segmentation, recognition, and post-processing steps will streamline large-scale manuscript analysis and facilitate integration with digital humanities platforms.

By pursuing these directions, future systems will be better equipped to handle the diversity and complexity of historical manuscripts, enabling more accurate, scalable, and accessible analysis of cultural heritage documents.

Chapter 4

Conclusion

This thesis presented a comprehensive investigation into semantic segmentation of point clouds for heritage buildings, as well as handwritten text extraction from historical manuscripts. Through a structured approach, we reviewed the state of the art, analyzed existing datasets, and developed both classical machine learning and deep learning pipelines tailored to the unique challenges of cultural heritage data.

For point cloud segmentation, we implemented and modified two leading neural network architectures, PointNet++ and DGCNN, adapting them to the ArCH dataset and heritage-specific requirements. Our experiments demonstrated that architectural modifications such as enhanced feature propagation, contextual layers, and tailored data preprocessing provided improvements in segmentation accuracy, in particular for underrepresented and geometrically complex classes. However, persistent challenges remain, notably data scarcity, class imbalance, and variable data quality. The limited size and diversity of annotated heritage datasets constrain model generalization, while noise and inconsistencies in point clouds further impact segmentation precision.

The classical ML approach, leveraging ensemble classifiers like Random Forests and Extra Trees, proved effective for propagating labels from partially segmented data, offering flexibility and reduced manual annotation workload. However, its performance is inherently limited by the completeness and representativeness of the initial labeled subset.

In the domain of handwritten text extraction, we surveyed existing AI platforms and open-source tools, highlighting the trade-offs between generic generative models and specialized ML pipelines. The analysis underscored the importance of annotated datasets, robust preprocessing, and custom architectures such as CRNNs and region-based segmentation networks for achieving reliable transcription and information extraction from historical manuscripts.

Our main findings confirm that while deep learning models can achieve promising results on structured heritage data, high-precision segmentation remains difficult without richer annotated datasets and further algorithmic advances. The integration of geometric consistency rules, ensemble methods, and domain adaptation strategies offers promising avenues for future improvement.

Looking ahead, the most impactful next steps include expanding annotated datasets, incorporating post-processing constraints to enforce architectural logic, and exploring

transfer learning and self-supervised techniques to mitigate data scarcity. For manuscript analysis, developing modular, end-to-end ML pipelines and leveraging multimodal information will further enhance accuracy and scalability.

The standalone program, developed as part of this work, provides a practical tool for heritage professionals enabling user-friendly segmentation and analysis of new point clouds. Its modular design and parameter flexibility lay the groundwork for future enhancements and broader application.

In summary, this thesis establishes a solid foundation for automated analysis of cultural heritage data, demonstrating both the potential and limitations of current ML and DL approaches. Continued research in dataset enrichment, model adaptation, and workflow integration will be essential to fully realize the benefits of AI in heritage conservation and historical document analysis.

Bibliography

[1] Sofia Ares Oliveira, Benoit Seguin, and Frederic Kaplan. dhsegment: A generic deep-learning approach for document segmentation. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 7–12. IEEE, aug 2018.

[2] Hassan El Bahi. Handwritten text recognition and information extraction from ancient manuscripts using deep convolutional and recurrent neural network. *Soft Computing*, 28(20):12249–12268, 2024.

[3] Catmus medieval dataset. https://huggingface.co/datasets/CATMuS/medieval. Accessed: 2025-08-28.

[4] Edgard Chammas, Chafic Mokbel, and Laurence Likforman-Sulem. Handwriting recognition of historical documents with few labeled data. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 43–48. IEEE, apr 2018.

[5] escriptorium: Platform for automatic transcription and annotation of historical documents. https://gitlab.com/scripta/escriptorium. Accessed: 2025-08-28.

[6] The esposalles database. https://dag.cvc.uab.es/dataset/the-esposalles-database/. Accessed: 2025-08-28.

[7] D. Griffiths and J. Boehm. Weighted point cloud augmentation for neural network training data class-imbalance. *arXiv preprint arXiv:1904.04094*, 2019.

[8] E. Grilli. *Automatic classification of architectural and archaeological 3D data*. PhD thesis, Alma Mater Studiorum, UniversitÃ di Bologna, Bologna, Italy, 2020.

[9] F. Matrone, A. Lingua, R. Pierdicca, E. S. Malinverni, M. Paolanti, E. Grilli, F. Remondino, A. Murtiyoso, and T. Landes. A benchmark for large-scale heritage point cloud semantic segmentation. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XLIII-B2-2020, pages 1419–1426, 2020.

[10] R. Pierdicca, M. Paolanti, F. Matrone, M. Martini, C. Morbidoni, E. S. Malinverni, E. Frontoni, and A. Lingua. Point cloud semantic segmentation using a deep learning framework for cultural heritage. *Remote Sensing*, 12(6):1005, March 2020.

[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[12] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[13] Adrià Simistierra, Alejandro H. Toselli, and Enrique Vidal. Carabela: A large-scale

dataset for handwritten text recognition of spanish documents written in the 16th–18th centuries, 2022.

[14] Transkribus: Platform for ai-powered text recognition of historical documents. https://www.transkribus.org. Accessed: 2025-08-28.

[15] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[16] Y. Yang, X. Wu, T. He, H. Zhao, and X. Liu. Sam3d: Segment anything in 3d scenes. *arXiv preprint arXiv:2306.03908*, 2023.

[17] H. Yue, Q. Wang, L. Huang, and M. Zhang. Enhancing point cloud semantic segmentation of building interiors through diffusion-based scene-level synthesis. *Automation in Construction*, 178:106390, 2025.