TUM School of Engineering and Design

Chair of Computational Modeling and Simulation
Prof. Dr.-Ing. André Borrmann

Door and Window Detection in Indoor Environment

February 2024

Report Submission

for

SOFTWARE LAB-2023 (BV030004)

RESEARCHER:

Isheanesu Roy Mugwagwa 03771519

SUPERVISORS:

Miguel Arturo Vega Torres

Mansour Mehranfar

Contents

1. Abstract

In this study, we address the challenge of object detection for doors and windows within indoor environments. Four major steps were considered in this project. Initially, a real-time object detecting Yolo V8 network was trained using an RGB image dataset, achieving an Average Precision (AP) of 89.5% on both RGB and point cloud image testing. Subsequently, another Yolo V8 network was trained using a hybrid dataset of point cloud and RGB images, yielding an AP of 92.2% on cross-testing these image types. Thirdly, a dedicated Yolo V8 network trained solely on point cloud data achieved an AP of 86.5% across RGB and point cloud image testing. This approach involved diverse dataset training to evaluate the model's adaptability and generalization to varied input data types. The point cloud images were obtained from projecting the point cloud data on to a BIM model and then take a screenshot of it. The RGB images were obtained from RealSense Camera. Moreover, to enhance the semantic understanding of detected elements, a classification based on material attributes, distinguishing between glass and non-glass elements, was implemented using MLP classification network, achieving an accuracy of 90%. Detection of metal or timber materials was feasible, although it required additional intensity information. This research significantly contributes to efficient indoor object detection methodologies, presenting potential applications in building automation and surveillance. Link to the code https://github.com/ishenroy/door_and_Window_detections

Keywords: object detection, “Building Information Modelling” (BIM), “Multilayer perceptron” MLP, RealSense Camera, Yolo v8, point cloud, RGB, real-time, intensity, building automation, semantic.

2. Introduction

2.1 Motivation

The door detection problem has been studied since the beginning of the 2000s and is still one of the hot topics in many research areas. [4] The main reason for that is the windows and doorway locations separate indoor environments into different sections. The ability to detect doors and windows via object detection algorithms facilitates several practical applications. For instance, being aware of a doorway location could promote the autonomy of robots and in building automation, efficient detection of these elements can aid in tasks like room segmentation, or occupancy monitoring. In the context of surveillance or security systems, accurate detection of doors and windows allows for better monitoring and tracking within indoor spaces. Moreover, the utilization of object detection techniques for doors and windows aligns with the growing interest in developing intelligent systems for smart homes, robotics, and assistive technologies. Enabling machines to recognize and understand such architectural features using visual data from cameras, such as RGB or depth images, contributes significantly to the advancement of autonomous navigation and smart environment technologies. In localization problem, doorways can be used as landmarks to decrease localization errors (Ehlers, Stuede, Nuelle, and Ortmaier, 2020). Determining doorway locations and traversing narrow passages may help automated wheelchair platforms and robots' fully autonomous navigation tasks (Kakillioglu, Ozcan, and Velipasalar, 2016; Derry and Argall, 2013). Also, classifying the status of the doors as open, semi-opened, and closed can improve the performance of humanoid robots for service tasks (Meeussen, Wise, Glaser, and Chitta, 2010). Besides, door detection is also crucial for Building Information Model (BIM) applications such as generating up to date models of large buildings (Jung, Stachniss, Ju, and Heo, 2018), for emergency cases, automatic extraction of IndoorGML models and segmentation of walkable spaces (Staats, Diakité, Voûte, and Zlatanova, 2019; Flikweert, Peters, Díaz-Vilariño, Voûte, and Staats, 2019), semantic mapping, and space subdivision (Nikoohemat, Peter, Elberink, and Vosselman, 2017; Zheng, Peter, Zhong, Oude Elberink, and Zhou, 2018).

2.2 Overview

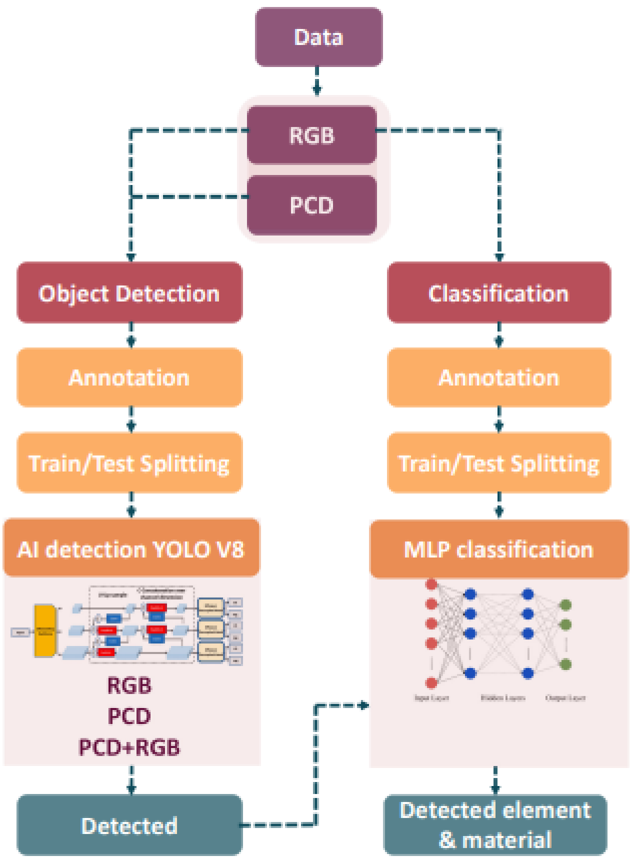

Figure 1 - Process workflow

As shown in the figure 1, the complete work mainly consists of 2 parts:

- Object detection using point cloud images and RGB images,
- Classification.

2.2.1 Hardware

The required hardware is listed below:

- Intel RealSense D435i stereo depth camera for the data acquisition.
- Iphone for capturing RGB images.
- Laptop: to obtain point cloud, RGB images and respective rough camera poses.

2.2.2 Software

- The alignment of the point cloud with the BIM model, to obtain the transformation matrix, is performed in CloudCompare.

- The majority of the code was programmed with Python using Google Collab.
- Yolo v8 model was used for object detection of doors and windows.
- MLP model was used for classification of Glass and Non-glass.
- A browser open source software called https://www.makesense.ai/ was used for annotations of the images.

3. Method

3.1 Object detection

3.1.1 Acquisition of dataset

- A total number of 350 RGB images was captured using Intel RealSense D435i stereo depth camera and Iphone camera.
- The RGB images comprised of Doors and Windows.
- We obtained 84 point cloud images by projecting point cloud data in a BIM model called CloudCompare and the we took a screenshot of it.

3.1.2 Data Sources

- The RGB images were captured from TUM main Campus in Building 1 and some were captured from TUM Garching Campus in Informatics building.

Figure 2 - RGB Images

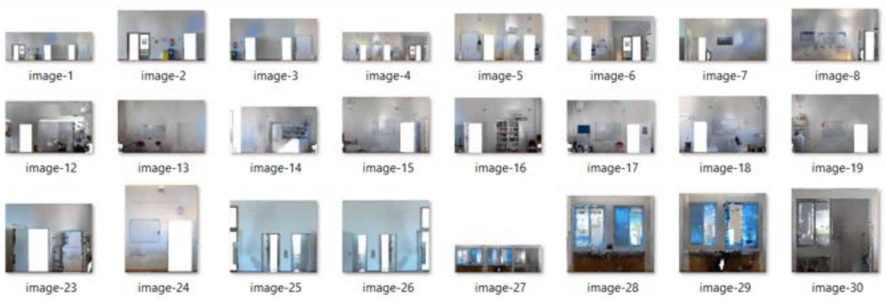

Figure 3 - Projected point cloud images

3.1.3 Annotation

- Software: A web-based software called makesense.ai was used to annotate the images.
- Labelling shape: The annotations were bounding boxes and class labels for the doors and windows.
- Labelling format: The labels were exported in YOLO format because YoloV8 was used for object detection. Labels in Yolo format is compatible with YoloV8 model.

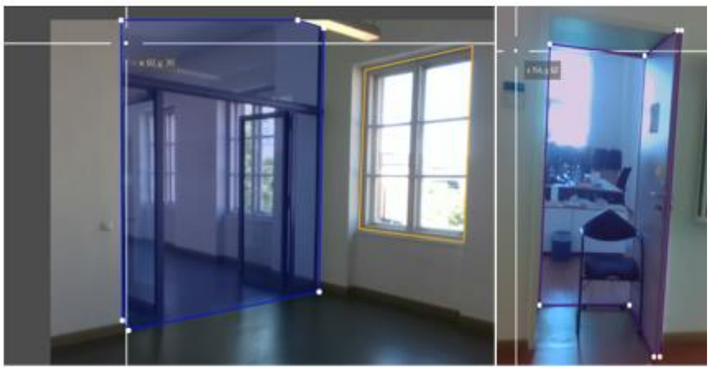
Figure 4 - Annotation of RGB images

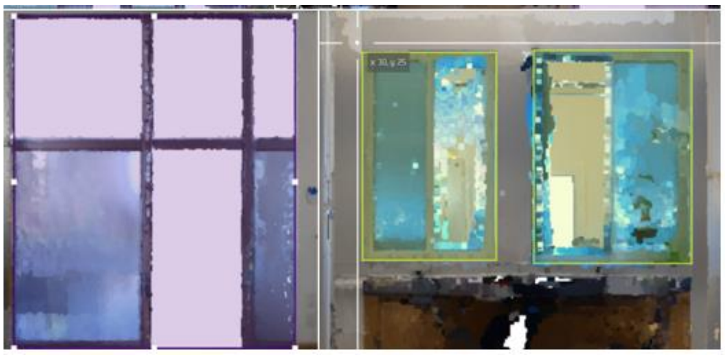
Figure 5 - Annotation of project point cloud images

3.2 Data Preparation

3.2.1 Data augmentation

Scaling and rotation data augmentation techniques were applied, to increase the diversity of the training data and improve model robustness.

3.2.2 Data splitting

For evaluation during and after training of the YoloV8 model, the dataset was split into training (70%), validation (20%), and test (10%) sets.

3.3 Network and Training

3.3.1 Visualization of the architecture of YoloV8

The YOLOv8 architecture makes use of a few key components to perform object detection tasks. The Backbone is a series of convolutional layers that extract relevant features from the input image. The SPPF layer and the subsequent convolution layers process features at a variety of scales, while the Upsample layers increase the resolution of the feature maps. The C2f module combines the high-level features with contextual information to improve detection accuracy. Finally, the Detection module uses a set of convolution and linear layers to map the high-dimensional features to the output bounding boxes and object classes. The overall architecture is designed to be fast and efficient, while still achieving high detection accuracy. As for the diagram legend, the rectangles represent layers, with the labels describing the type of layer (Conv, Upsample, etc.) and any relevant parameters (kernel size, number of channels, etc.). The arrows represent data flow between layers, with the direction of the arrow indicating the flow of data from one layer to the next. [5]

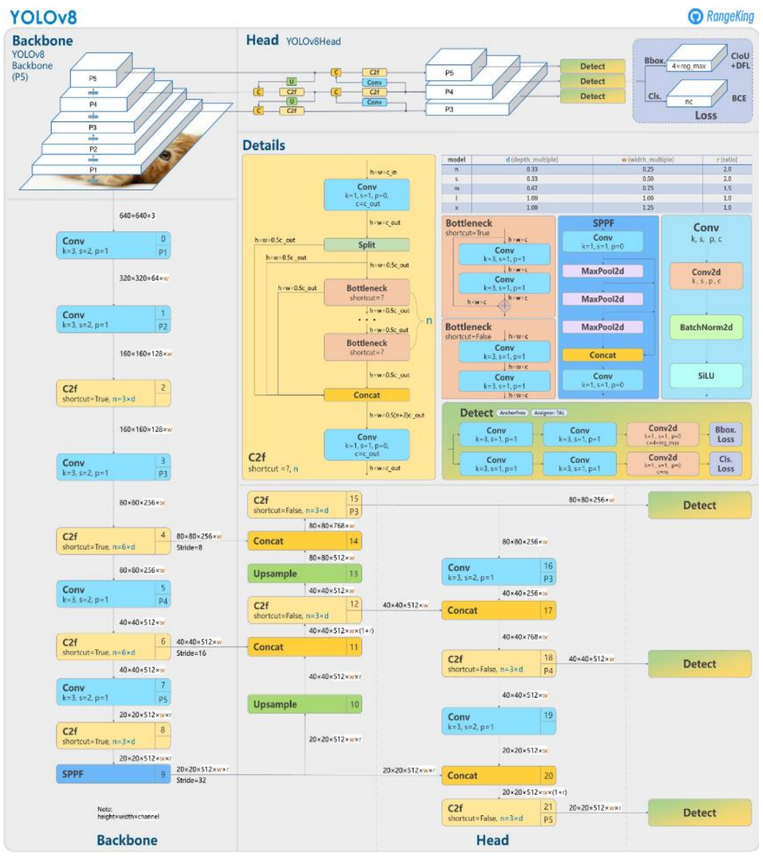

Figure 6 - YOLO V8 Architecture (ref : ....)

3.3.2 Comparison with other architectures

Table 1 - Comparison between architectures

		Mask R-CNN	YOLOv8
1	Architecture	Extends the Faster R-CNN architecture by adding a parallel mask prediction branch.	Uses a single neural network to directly predict bounding boxes and class probabilities.
2	Speed and Efficiency	Relatively slower inference speed due to multi-stage architecture.	Real-time detection capabilities, offers faster inference speeds
3	Detection Performance	Good instance segmentation performance, producing accurate segmentation masks	Designed for real-time object detection and offers faster inference speed
4	Training and Dataset Requirements	Typically requires more training time due to its multi-stage architecture and the additional mask prediction branch.	Trains faster compared to Mask R-CNN due to its single-stage architecture.

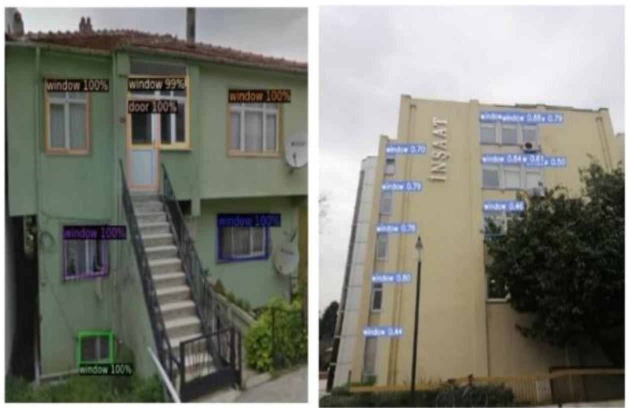

Figure 7 - Door/window detection using Faster R-CNN architecture (ref : : *Deep Learning –based door and window detection from Building)Façade)*

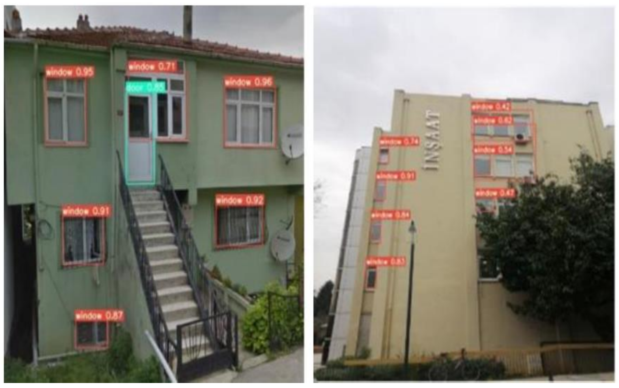

Figure 8 - Door/window detection using YOLOv8 architecture (ref : : *Deep Learning –based door and window detection from Building)Façade)*

3.3.3 Data processing using YoloV8

Following are the basic steps of the working principle of YOLO v8:

a) Input Processing: YOLO v8 takes an image as input and divides it into a grid, typically with a size of 13x13 or 26x26, depending on the specific type. Each grid cell is responsible for predicting objects within its spatial domain.
b) Feature Extraction: The network extracts high-level features from the input image using a deep convolutional neural network (CNN). The choice of the network architecture often taken from established models like ResNeXt or Darknet.
c) Bounding Box Prediction: YOLO v8 predicts the bounding box for objects by regressing the coordinates of the top-left corner, width and height of the box. Additionally, it calculates a confidence score that indicates the probability that the predicted box contains the object.
d) Class Prediction: Along with bounding box predictions, YOLO v8 also predicts class probabilities for each grid cell. This means that the model can not only detect objects but also identify their relative categories.
e) Post-Processing: Once the predictions are made, a confidence threshold is applied to filter out low-confidence detections. Non-maximum suppression is then used to remove duplicate or overlapping bounding boxes, ensuring that only the most accurate search continues.
f) Loss Function: YOLOv8 typically uses a combination of loss functions, including:

- Classification Loss: To predict object class probabilities.
- Localization Loss: To predict bounding box coordinates accurately.
- Confidence Loss: To estimate object scores, penalizing false positives and negatives.
- These loss functions are designed to optimize the model's performance in object detection tasks.

YOLOv8 is well-suited for indoor door and window detection due to its real-time processing capabilities, efficient single-stage architecture, and the ability to handle objects of varying sizes and orientations. The use of anchor boxes helps in localizing objects accurately, making it suitable for detecting doors and windows in indoor environments.

3.3.4 Training

a. **Hyperparameters:** Training hyperparameters like learning rate (0.001), batch size (8), Epoch (150), Input channels (RBG) and optimizer (Adam Weight Decay). These parameters are essential because they determine how the model learns from data.

b. **Training Process**: The model was trained using stochastic gradient descent (SGD). During training the model iteratively updated its weights to minimize the combined loss function across the training data. Once the training was complete, the model was capable of accurately detecting most doors and windows in new images.

Three YoloV8 models with the same hyperparameters as mentioned above where trained and tested using different datasets:

1) **First model** was trained using RGB images and then tested on RGB images and Point cloud images.
2) **Second model** was trained using Point cloud images and RGB images and then tested on Point cloud images.
3) **Third model** was trained using Point cloud images and was tested on Point cloud images and RGB.

3.3.5 Metrics Of Accuracy

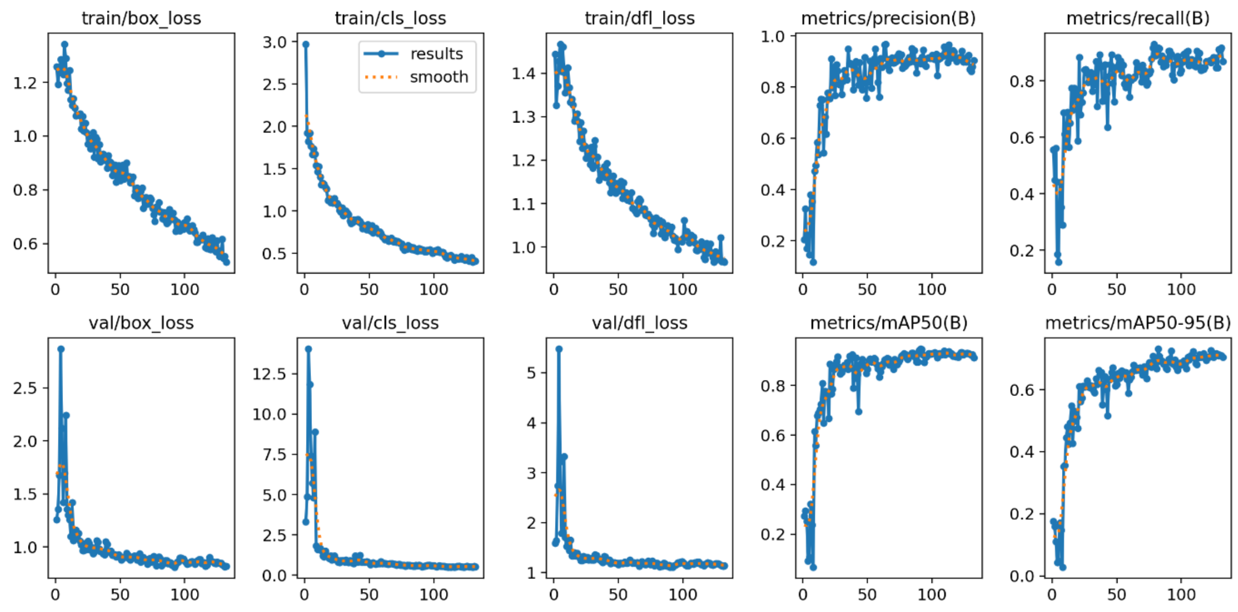

Figure 9 – Metrics of Accuracy

a. **Train/box_loss graph**- measures the discrepancy between predicted and actual bounding box coordinates during training, reflecting the model's improvement in accurately localizing objects in images. A decreasing trend in the box loss graph signifies that the model is gradually improving in its ability to predict bounding box coordinates accurately.
b. **Val/box_loss graph** illustrates the model's capability to precisely locate Doors and windows in images during validation. A decreasing trend signifies the model's improving accuracy in predicting the bounding box coordinates of doors and windows on unseen validation data.
c. **Train/cls_loss graph** evaluates how well the model learns to assign correct labels 'door' or 'window' to the detected objects during training. A decreasing trend indicates the model is improving in accurately classifying doors and windows, while fluctuations or an increasing trend may suggest challenges in learning to classify these objects, potentially due to factors like data imbalance, complexity in distinguishing classes, or overfitting.
d. **val/cls_loss graph** evaluates how accurately the model categorizes detected doors and windows, on a separate validation dataset throughout the training process. A decreasing trend in the graph indicates improving accuracy in classifying doors and windows on unseen validation data.
e. **Train/dfl_loss:** This represents the loss associated with the detection confidence of objects (objectness score) during the training process. A decreasing trend indicates that the model is improving in predicting whether an object is present in a certain region of the image.
f. **Val/dfl_loss:** Like the "Train/dfl_loss" but measured during validation. It shows how well the model predicts objectness scores on unseen validation data. A decreasing trend suggests the model's ability to generalize in determining whether doors and windows, are present in specific image regions.
g. **Metrics/Precision (B):** Precision is a metric that measures the accuracy of positive predictions made by the model. Here, (B) refer to a specific class such as doors or windows. It shows the precision of the model in correctly identifying instances of these objects among all predicted instances.
h. **Metrics/mAP50(B):** mAP (mean Average Precision) at 50% intersection-over-union (IoU) is an evaluation metric that calculates the average precision of the model at a certain IoU threshold (here, 50%). This metric assesses the overall performance of the

model in detecting specific objects, like doors and windows, considering both precision and recall.

i. **Metrics/Recall(B):** Recall measures the model's ability to correctly detect all instances of a particular class (B) among all the actual instances present in the dataset. A higher recall indicates that the model is effectively capturing most instances of the specified objects.

j. **Metrics/mAP50-95(B):** This metric calculates the mean Average Precision across different IoU thresholds (from 50% to 95%) for a specific class (B). It provides a comprehensive evaluation of the model's detection performance across a range of IoU thresholds for the specified objects.

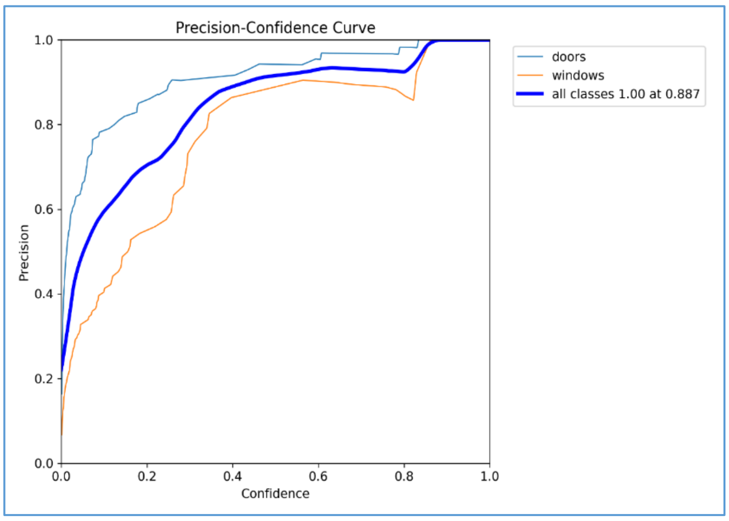

Figure 10 - Precision - Confidence Curve

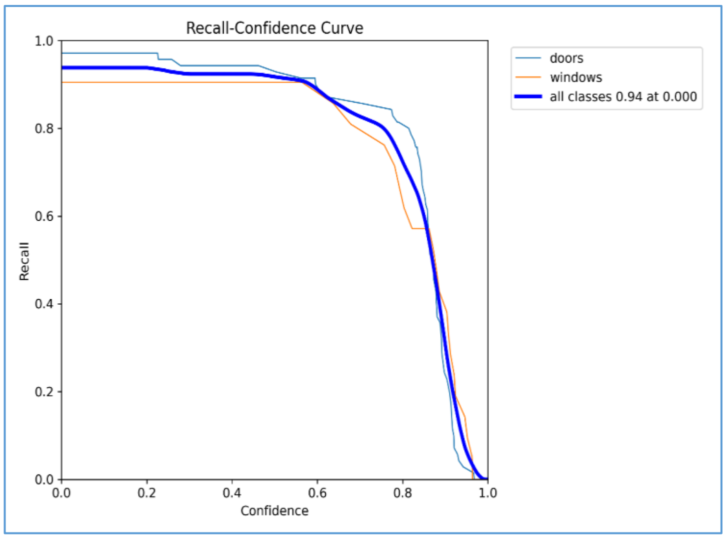

Figure 11 – Recall Confidence Curve

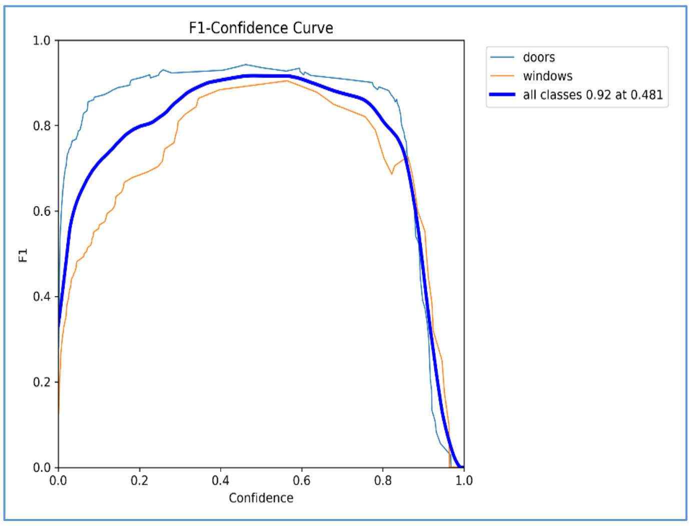

Figure 12 - Confidence Curve

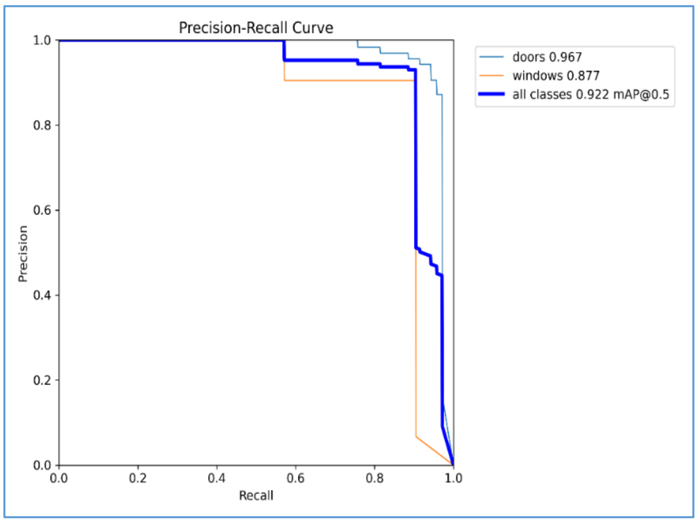

Figure 13 - Precision Recall Curve

The precision-recall curve shows the trade-off between precision and recall for different thresholds. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

$$\boldsymbol{Precision = \frac{TP}{TP+FP}} \tag{1}$$

$$\boldsymbol{Recall = \frac{TP}{TP+FN}} \tag{2}$$

$$\boldsymbol{F - Score = 2\,\frac{Precision * Recall}{Precision + Recall}} \tag{3}$$

Where :

TP = true positives

FP= false positives

FN = false negatives

$$AP = \sum_n (R_n - R_{n-1}) P_n \qquad (4)$$

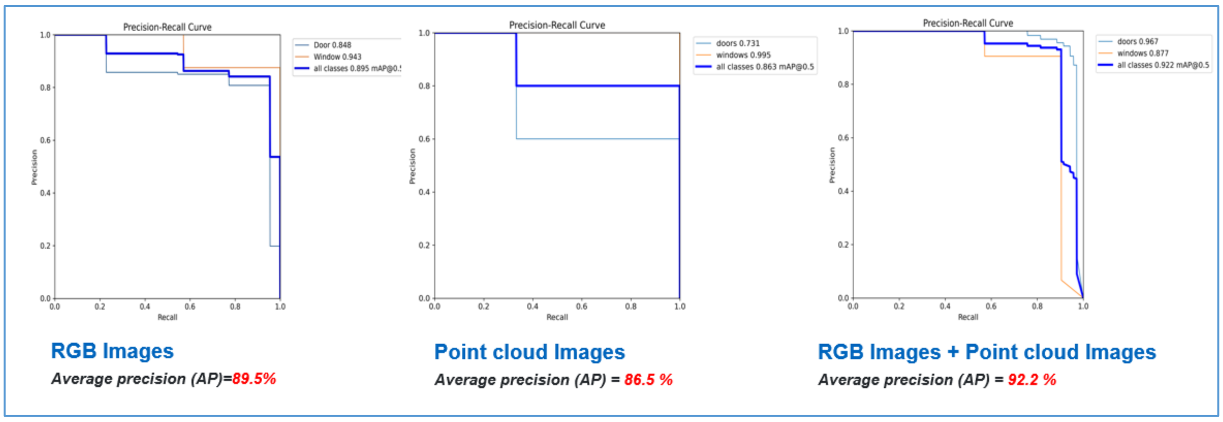

Figure 14 – Comparison result

The graphs achieved different Average precisions and different Precision-Recall Curve due:

3.3.6 Data Relevance and Representation:

The first model trained on RGB images perform well on RGB test data due to familiarity with the visual spectrum. However, its performance might decrease when tested on Point cloud images as it lacks direct exposure to Point cloud representations during training.

The second model trained on both RGB and Point cloud images exhibit higher performance on Point cloud images during testing because it learned features from both data modalities during training, allowing better adaptability to Point cloud representations.

The third model trained exclusively on Point cloud images resulted in slightly lower performance on both Point cloud and RGB test data. This model lack exposure to RGB data during training, affecting its ability to generalize to RGB images during testing.

3.3.7 Feature Learning and Generalization:

Models trained on diverse datasets have varying capabilities to extract features and generalize across different data modalities. A model trained on multiple modalities learn more diverse and robust features, contributing to better performance on both modalities during testing.

a. **Domain-Specific Characteristics:**
   Point cloud data has distinct characteristics (such as 3D spatial information) compared to RGB images (which contain colour information in 2D space). Models specialized in processing specific data types outperform others when tested on similar data.

b. **Model Adaptability and Robustness:**
   The adaptability and robustness of a model to handle different data representations significantly influence its performance. Models that effectively adapt to diverse data types during training tend to generalize better and perform well across various testing scenarios.

3.3.8 Test Results

Model 1 (Training on RGB images and testing on RGB Images + point cloud images)

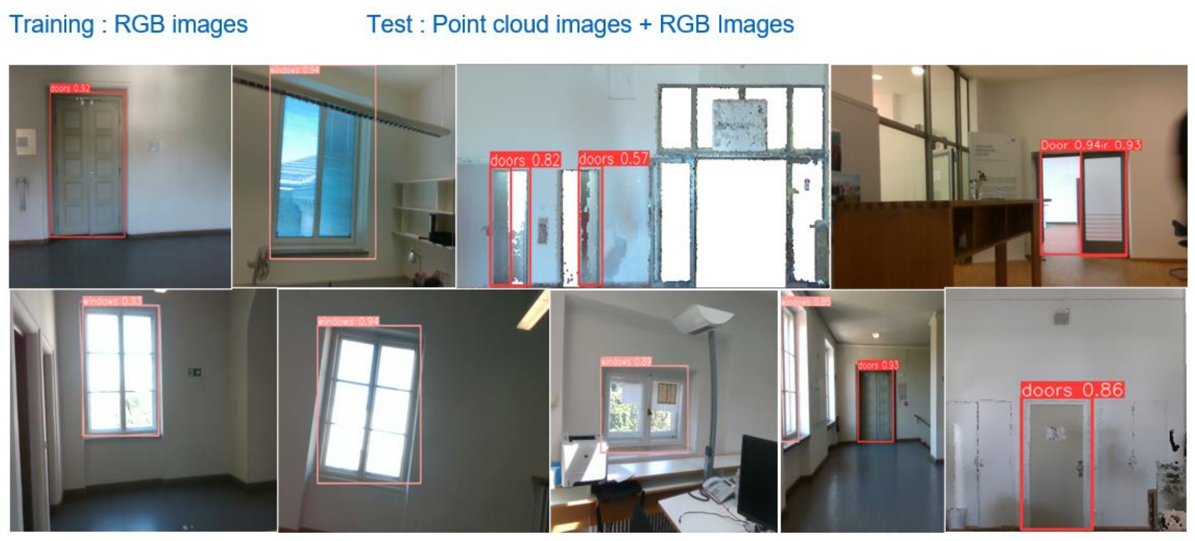

Figure 15 - Model - RGB images, Training - RGB + point cloud images

The model performed well in detecting RGB images but it did not detect some point cloud images because RGB images and point cloud data present information differently. The features learned from RGB images might not effectively generalize to point cloud data, leading to a mismatch between the expected features for object detection and those present in point cloud images.

Model 2 (Training on RGB images + point cloud images and testing on point cloud images)

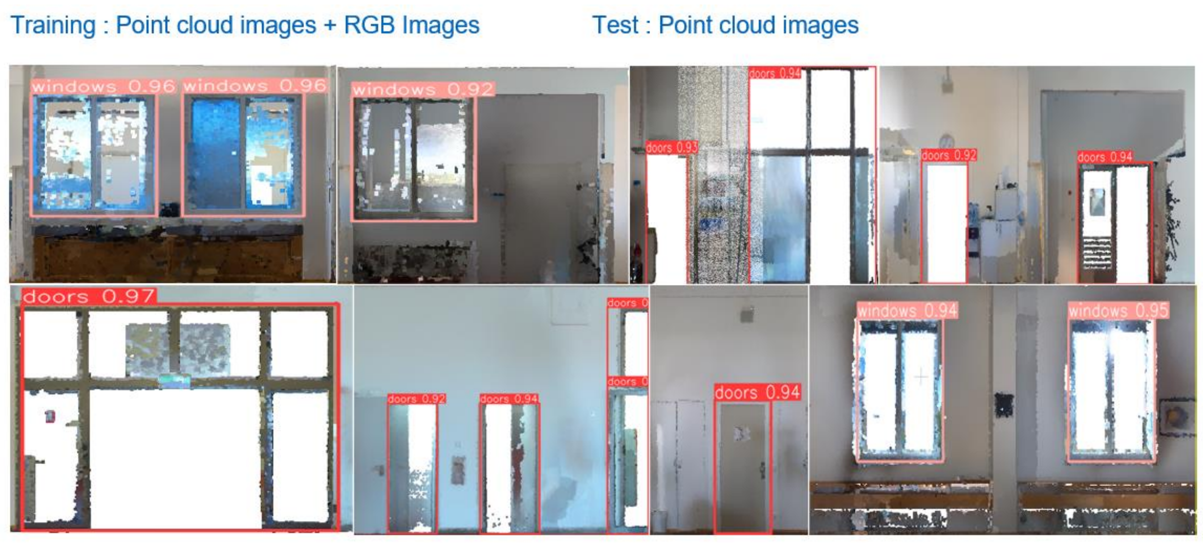

Figure 16 - Model - point cloud images + RGB images, Testing - point cloud images

The second model was trained on a combination of RGB images and point cloud images, allowing it to learn features from both modalities. When tested on point cloud images, this model demonstrated higher performance for several reasons:

Larger Dataset: It increases the amount of the data set and its diversity which leads to that network learn different shape, situation and scenario.

Modality Fusion during Training: By incorporating both RGB images and point cloud data during training, the model had exposure to and learned features from both modalities. This fusion allowed the model to capture diverse information present in both types of data representations, enhancing its ability to detect objects within point cloud images.

Enhanced Feature Representation: Training on a combination of RGB and point cloud data potentially enriched the model's feature representations. It gained insights into spatial information from point cloud images while also leveraging the colour and texture information from RGB images, leading to a more comprehensive understanding of object characteristics.

Model 3 (Training on point cloud images and testing on RGB images + point cloud images)

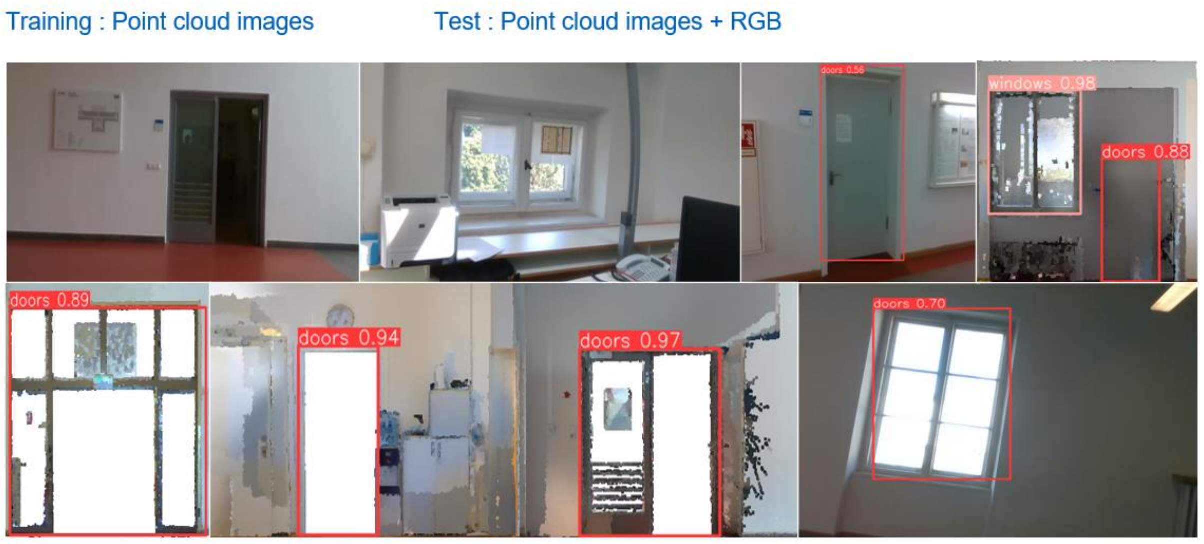

Figure 17 - Model- point cloud images, testing - point cloud + RGB images

The third model was trained exclusively on point cloud images, which offer a three-dimensional representation of spatial information. While it performed well in detecting objects within point cloud images, its relatively lower performance in detecting RGB images because the model was trained solely on point cloud data, lacking exposure to RGB images during training. As a result, the learned features and representations might be specific to the characteristics of point cloud data, such as spatial relationships, density, or 3D structures.

The features and patterns learned from point cloud images might not directly translate or generalize well to the different representation of RGB images. RGB images provide colour and texture information in a two-dimensional format, which differs significantly from the 3D spatial information present in point cloud data.

3.4 Classification

3.4.1 Image Acquisition

We detect the objects in Point cloud images and RGB images and now we want to enrich the semantic information of detected elements by classifying them based on their materials into glass and non-glass. For metal or timber materials, it’s possible to detect, but we need additional information like intensity. This can be a future scope.

Dataset: A dataset of 125 images was used. The non-glass images were either timber or metal with different colours.

Data Sources: The RGB images were captured from TUM main Campus in Building 1 and some were captured from TUM Garching Campus in Informatics building

Class: ‘None Glass’

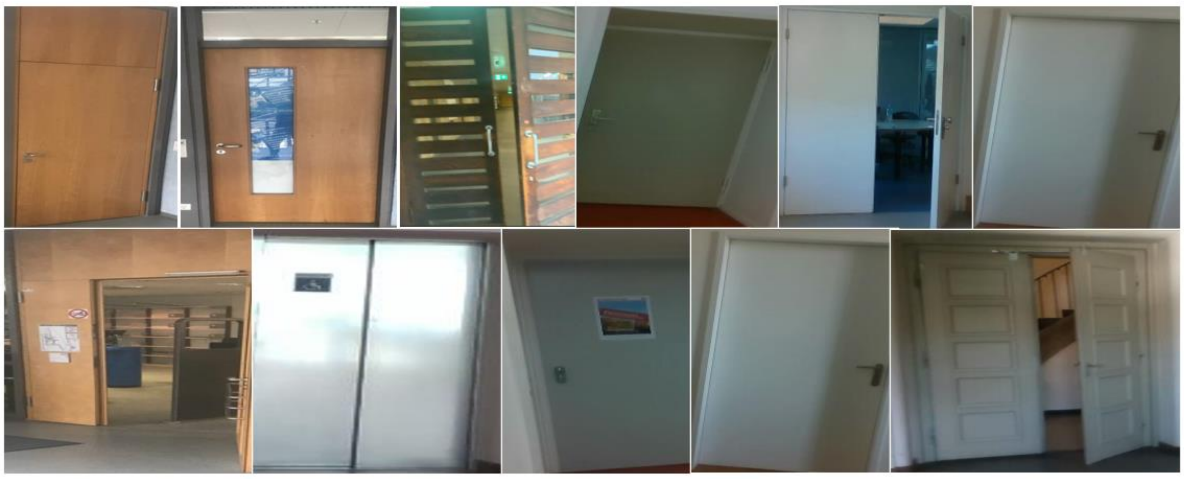

Figure 18 - Class "None Glass"

Class: ‘Glass’

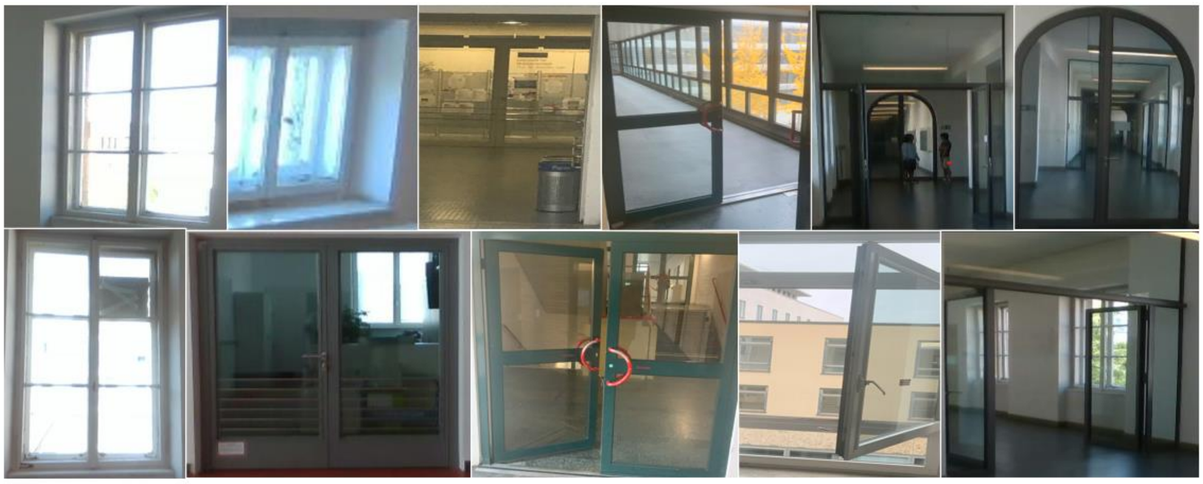

Figure 19 - Class "Glass"

3.4.2 Training

a. Network setting and Hyper Parameters

Multilayer Perceptron (MLP) network was used for classification. MLP network is comprised of one or more layers of neurons. Data is fed to the input layer, there may be one or more hidden layers providing levels of abstraction, and predictions are made on the output layer, also called the visible layer. MLPs are suitable for classification prediction problems where inputs are assigned a class or label so that is why we used the network in this project. [7]

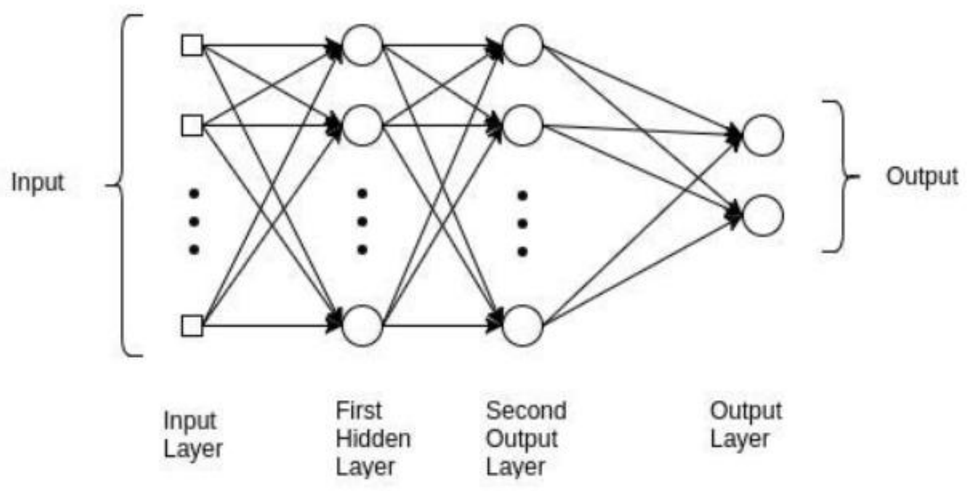

Figure 20 - MLP network (ref: analayticsvidhya.com)

b. Training Configuration:

- **Batch Size:** The number of samples propagated through the network during each training iteration was set to 8.
- **Epochs:** The training process iterated through the entire dataset 85 times (epochs) to update the network's weights and improve its classification accuracy.
- **Image Size:** The images used for training and inference were resized to 100x100 pixels.
- **Learning Rate:** The learning rate for the Adam optimizer was set to 115ms/epoch, controlling the step size taken during optimization to update the network's parameters.
- **Optimizer:** The Adam optimizer, a popular optimization algorithm in deep learning, was employed to update the network's weights iteratively based on the training data, aiming to minimize the classification error and improve accuracy.

- **Accuracy:** The model achieved an accuracy of 90% on the validation or test set, indicating that it correctly classified 90% of the doors and windows into their respective glass and non-glass categories.

c. Confusion Matrix:

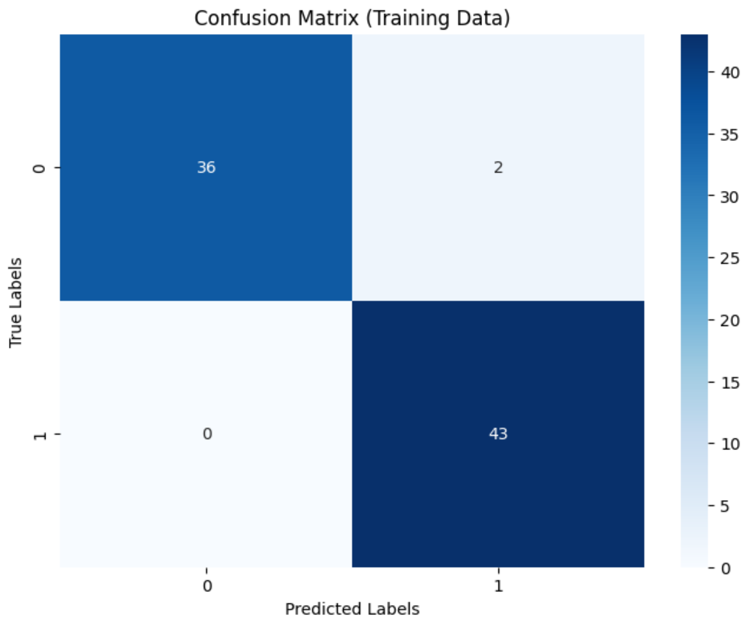

Figure 21 - Confusion Matrix

This matrix allows us to assess the performance of your model in terms of correctly and incorrectly classifying 'Glass' and 'Non-glass' instances.

The diagonal elements (top-left to bottom-right) represent correct predictions. Off-diagonal elements represent incorrect predictions. The row-wise sums give the total number of instances for each actual class. The column-wise sums give the total number of instances for each predicted class.

3.4.3 Results

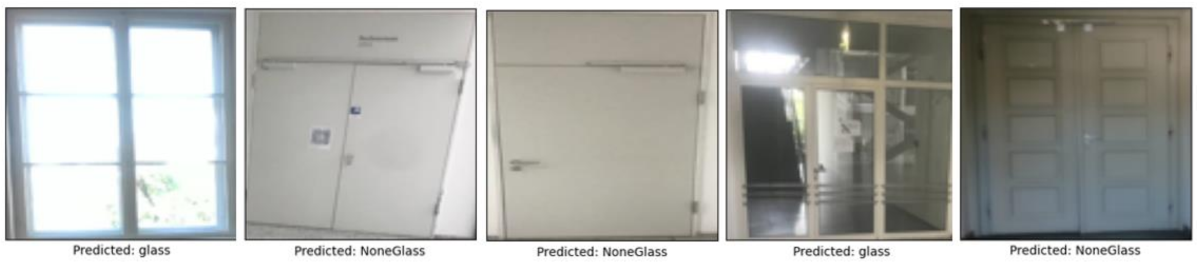

Figure 22 - Classification results

The network performed well in classifying the glass and non-glass due to

Representation Learning: The network's architecture allowed it to learn and extract relevant features from the input images. By utilizing multiple layers of neurons, the MLP could capture hierarchical representations that helped distinguish between glass and non-glass elements in the doors and windows.

Rich Information from RGB Channels: Leveraging the RGB channels in the input data enabled the model to capture colour-based information crucial for distinguishing glass from non-glass surfaces. The network learned to recognize patterns and features associated with the appearance of glass materials in RGB images.

4. Conclusion and Scope of Future Work

Effectiveness of Combining RGB and Projected Point Cloud Images: The achieved average precision of 92.2% in detecting door and window elements highlights the efficacy of utilizing a combination of RGB images and projected point cloud (PCD) images for training. The fusion of these modalities likely provided a more comprehensive and diverse set of visual information, enabling the detection network to learn and generalize better, leading to a higher average precision.

- Classification Performance based on Material (Glass and Non-glass): The overall accuracy of 90.00% in classifying materials into glass and non-glass categories demonstrates a strong performance of the classification model. This high accuracy indicates the model's proficiency in distinguishing between glass and non-glass materials, likely leveraging distinct visual cues present in the provided dataset.
- Limitation in Object Detection: Rectangular Shapes: The object detection network's limitation in detecting only doors and windows in rectangular shapes signifies a potential constraint in the model's ability to recognize non-rectangular or irregularly shaped elements. This limitation might stem from the network architecture, or the characteristics of the dataset used for training, indicating a need for further model enhancements to detect a broader range of shapes.
- Challenges in Glass Material Classification due to Background Colour: The classification model encountered difficulties in accurately classifying some glass

materials, possibly due to colour backgrounds present in the dataset. The presence of similar colours or textures in the background might have caused confusion for the model, affecting its ability to precisely classify certain glass materials.

Future work suggestions

- Enhance object detection to recognize diverse shapes beyond rectangles, exploring advanced architectures and additional dataset diversity.
- Integrating segmentation techniques into future work can potentially enhance object recognition by precisely delineating object boundaries, enabling finer object characterization and improving the accuracy of identifying and classifying door and window elements, especially in scenarios where the shapes are more complex or irregular.
- Improve the glass material classification model by addressing colour background complexities, potentially through background removal or diverse training data.
- Fine-tune parameters and architectures to refine both detection and classification models for increased accuracy and robustness.

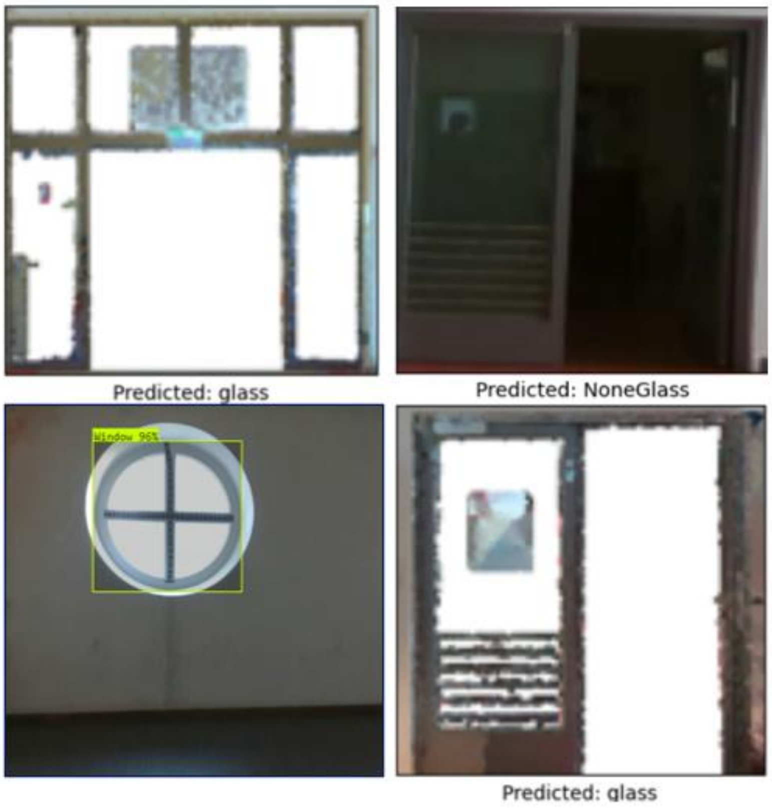

Figure 23 - Images showing limitations of work

Figures

Table

Bibliography

[1] K. Asadi et al. “Real-Time Image Localization and Registration with BIM Using Perspective Alignment for Indoor Monitoring of Construction”.

In: Journal of Computing in Civil Engineering 33(5) (2019).

DOI: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000847.

[3] R. Bowen, E. Cinar, and F. Sahin. “System of systems approach to a human tracking problem with mobile robots using a single security camera”.

In: 2009 Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control (2009).

DOI: https://doi.org/10.1109/ICSCCW.2009.5379435.

[8] V. Hedau, D. Hoiem, and D. Forsyth.

"Recovering the Spatial Layout of Cluttered Rooms".

In: Twelfth IEEE International Conference on Computer Vision (2009).

[2] "Position Detection of Doors and Windows Based on DSPP-YOLO"

Tong Zhang, Jiaqi Li, Yilei Jiang, Mengqi Zeng 1 and Minghui Pang

DOI: https://www.mdpi.com/2076-3417/12/21/10770

[4] "COMPARISON OF DEEP LEARNING TECHNIQUES FOR DETECTION OF DOORS IN INDOOR ENVIRONMENTS"

Burak Kaleci and Kaya Turgut

DOI: https://www.researchgate.net/publication/356973212_COMPARISON_OF_DEEP_LEARNING_TECHNIQUES_FOR_DETECTION_OF_DOORS_IN_INDOOR_ENVIRONMENTS

[5] "Brief summary of YOLOv8 model structure"

DOI: https://github.com/ultralytics/ultralytics/issues/189

[6] "A Simple overview of Multilayer Perceptron(MLP)"

DOI: https://www.analyticsvidhya.com/blog/2020/12/mlp-multilayer-perceptron-simple-overview/

[7] "When to Use MLP, CNN, and RNN Neural Networks"

DOI: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/

by Jason Brownlee