JÖNKÖPING UNIVERSITY
School of Engineering

Automated SQL Command Generation from Digital Diagrams Using Computer Vision Techniques

PAPER WITHIN *Artificial Intelligence*
AUTHORS: *Gizem Yilmaz, Adarsh Johny*
TUTOR: *Ulf Johansson*
JÖNKÖPING *May 2025*

This exam work has been carried out at the School of Engineering in Jönköping in the subject area Computer Science. The work is a part of the Two-year Master of Science in AI Engineering programme. The authors take full responsibility for opinions, conclusions and findings presented.
Examiner: Vladimir Tarasov
Supervisor: Ulf Johansson
Scope: 30 credits
Date: 2025-06-15

Mailing address:
Box 1026
551 11 Jönköping

Visiting address:
Gjuterigatan 5

Phone:
036-10 10 00 (vx)

Abstract

The development of warehouse automation systems frequently encounters obstacles when CAD based warehouse layouts are manually converted into database structures. This procedure slows development and deployment in logistical settings since it is laborious, prone to mistakes, and not scalable. To overcome this difficulty, this thesis, carried out in partnership with Consafe Logistics (CL), creates a Proof Of Concept (POC) for an AI driven pipeline that generates structured SQL commands from warehouse CAD designs.

The goal is to put in place a semi-automatic system that can identify essential layout elements and their positions to generate SQL statements that define driveways, and racks using YOLO models. From a starting set of 30 images, CAD layouts were annotated and artificially extended utilizing cropping, rotation, and flipping to create enough samples. These were used to train detection models to distinguish between driveways and racks and output the pixel coordinates.

The model trained on pre-trained YOLOV9c produces nearly 95+% positive results while the images are marked with color-coded reference cues for easy detection. These pixel positions are then converted to actual positions using a predefined scale in the drawing. These would potentially save around 50 hours of human work for one CAD drawing,

This thesis shows how deep learning techniques can be used to detect accurate locations of objects in engineering drawings. It boosts schema dependability, drastically reduces human labor, and helps warehouse operations transition to digital. Subsequent research will assess generalization and extend to different digital drawings and layout types other than warehouses.

Keywords: Warehouse Management System, CAD Automation, SQL Generation, Object Detection, YOLO, Computer Vision, Logistics AI, Digital Drawing

Contents

List of Figures

List of Tables

Acronyms

AdamW Adaptive Moment Estimation with Weight Decay. 32

AGV Automated Guided Vehicle. 5

AI Artificial Intelligence. 1–3, 6, 7, 14, 16, 18, 19, 40, 42, 49, I, II

BCE Binary Cross-Entropy. 12, 13, 25

BIM Building Information Modeling. 42, 43

CAD Computer-Aided Design. 1–33, 35–40, 42, 43, 49, I, II, IV

CL Consafe Logistics. 6, 18–21, 30, 49, I

CNN Convolutional Neural Network. 7

COCO Common Objects in Context. 3, 11–13, 29, IV

CPU Central Processing Unit. 29, 32, IV

CUDA Compute Unified Device Architecture. 26, 27

CV Computer Vision. 1, 2, 6–8, 13, 15, 18, 21, 40, 42, I

CVAT Computer Vision Annotation Tool. 3, 9, 18, 21, 31, II

DFL Distance Focal Loss. 25

DWG Drawing (AutoCAD file format). 19

FLOPs Floating Point Operations. 12

FPN Feature Pyramid Network. 9

GIS Geographic Information Systems. 15

GPU Graphics Processing Unit. 11, 19, 24, 26, 27, 29, 38

IoT Internet of Things. 6

IoU Intersection over Union. 11, 13

JSON JavaScript Object Notation. 26, 27, 30

mAP mean Average Precision. 11–13, 18, 29, 36, 42, IV

NMS Non-Maximum Suppression. 11

PDF Portable Document Format. 6

PGI Programmable Gradient Information. 11

PNG Portable Network Graphics. 6, 9

POC Proof Of Concept. I

R-CNN Region-based Convolutional Neural Network. 6, 7, 11

RQ Research Question. 1, 31–33, 39

SQL Structured Query Language. 1–4, 6, 14, 15, 17–19, 21, 23, 26–28, 30, 35–37, 39, 41–43, 49, I, III

VFL Varifocal Loss. 12, 13, 25, 26

WMS Warehouse Management System. 2, 4–6, 40, 42, 49, I, II

YOLO You Only Look Once. 1, 3, 4, 6, 7, 10–15, 18, 19, 23–27, 29, 32, 33, 36, 38, 39, 42, 43, I–IV

1 Introduction

The recent rise of artificial intelligence technologies has made automation a key factor in improving efficiency and accuracy across many industries. Populating database tables in warehouse management presents a significant challenge because it traditionally requires extensive manual effort. The operation of warehouses depends on complex Computer-Aided Design (CAD) diagrams, which show storage layouts, transportation paths, and inventory distribution. Turning graphical warehouse representations into structured databases through manual methods takes too much time and produces numerous errors.

This research explores a new method that transforms CAD diagram information into Structured Query Language (SQL) commands to populate warehouse databases automatically with minimal human effort. The research combines Computer Vision (CV) with human efforts to connect physical warehouse designs with digital databases, which enhances data management efficiency and reduces human errors.

1.1 Problem Statement

The process of designing a warehouse management database schema demands both data modeling expertise and logistics knowledge. CAD diagrams visually present key warehouse elements such as shelving layouts, storage units, and transportation routes. The process of extracting structured information from graphical representations remains challenging because of the complexity and variability of warehouse designs.

Current methods rely heavily on manual interpretation and textual specifications, which are insufficient for capturing the spatial relationships embedded in CAD drawings. This manual process is time-consuming and prone to inconsistencies and human errors that can lead to inefficiencies and warehouse mismanagement.

This thesis proposes a semi-automated pipeline combining computer vision techniques, structured database logic that converts CAD-based warehouse diagrams into SQL commands, with minimal human effort. The recent developments in object detection with YOLO enable accurate and efficient component extraction from structured CAD diagrams (Ultralytics, 2023, 2024a, 2024b; C. Wang et al., 2024). The proposed AI-assisted CAD-to-database conversion approach increases accuracy, reduces manual effort, and enables the creation of consistent digital twins from existing warehouse layouts.

1.2 Research Questions

The primary research question this thesis aims to answer is:

- **RQ1.** How can Computer Vision techniques be used to identify and extract spatial information from CAD diagrams?

To explore this question in more depth, the following sub-questions are formulated:

- **RQ1.1.** What Computer Vision techniques can be used for detecting key warehouse components in CAD-based layouts?
- **RQ1.2.** How can spatial relationships and pixel coordinates of detected elements be accurately extracted from annotated CAD images?

1.3 Purpose

The primary purpose of this thesis is to present a systematic, AI-driven approach to automating the extraction of spatial and semantic information from CAD models to populate existing warehouse database tables. In addition to standardizing warehouse database design and reducing the need for manual interpretation, this project aims to increase productivity by reducing human error. By examining current methods combining Computer Vision (CV) and SQL generation, it is aimed to contribute to the academic literature and to make progress in applied fields such as supply chain automation, logistics, and warehouse management. This research aims to shed some light on how we can extract digital drawing information into structured output using artificial intelligence.

1.4 Objectives

To answer the research questions (see 1.2), this thesis aims to develop an AI-based system using advanced Computer Vision (CV) techniques to accurately identify and classify important warehouse components such as storage units, racks, driveways, and similar items from warehouse CAD designs. The system will use semi-automated methods to precisely locate these components in the CAD drawings by determining their pixel-level positions. These positions will then be converted into real-world coordinates using the scales provided in the CAD files, along with relevant mathematical calculations.

Additionally, the thesis will create a semi-automated pipeline to convert the spatial and semantic data gathered into structured SQL scripts. This automation will streamline the population of warehouse databases with accurate component locations. A prototype system will be developed and tested to demonstrate the practicality and effectiveness of this automated CAD-to-SQL conversion process. Feedback from stakeholders and ongoing evaluations will be used to refine this prototype, This will ensure it is practical and reliable for real-world applications. This will reduce manual effort and save time, specifically during the setup phase of digital twin systems for warehouse environments, which is a bottleneck now.

As the final stage of this research, a comprehensive analysis and evaluation will be conducted by the thesis to determine the overall project effectiveness, scalability, and accuracy of the proposed system. Real-world case studies and comparisons with existing manual techniques will be a part of this evaluation. The applicability of the automated technique in real-world warehouse management scenarios will be evaluated by analyzing performance parameters, including accuracy, recall, execution time, and scalability. An expert will also be consulted during the evaluation phase to obtain more detailed opinions. By achieving these objectives, this thesis hopes to bridge the gap between the structured data format of digital drawing by providing a new and effective approach to warehouse database automation.

1.5 Thesis Structure

This report of the thesis is organized into six chapters, with each providing an in-depth analysis of the research problem, the AI-based solution created to address it, and the analysis of the results.

Chapter 1 identifies the problem statement, formulates the research questions, and explains the objectives and contributions of this thesis. The chapter sets the context by explaining the shortcomings that have been associated with the manual conversion of warehouse CAD floor plans to structured data and introduces the motivation for employing deep learning-based Computer Vision (CV) models for automating the process.

Chapter 2 provides background information and reviews pertinent literature. It discusses the use of relational databases in Warehouse Management System (WMS), the significance and difficulty of parsing

CAD layouts, and the development of object detection frameworks like YOLO. Additionally, it discusses annotation tools and formats like CVAT and YOLO/COCO and concludes with an analysis of gaps in existing research, namely, the absence of end-to-end pipelines for automating CAD to structured data workflows in warehouses.

Chapter 3 covers the methodology and implementation details of the proposed pipeline. It describes how to obtain and prepare CAD diagrams, how to establish the dataset through point annotations, and how to transform the annotated data into formats appropriate for training and inference. It also outlines how YOLO is trained for object detection, as well as the post-processing steps and the regulations employed to generate SQL queries from the output of the model.

Chapter 4 illustrates the results of the system. It includes experimental outputs, model performance, and example SQL generation outputs. It contains graphical examples and illustrates trained model performance in finding spatial elements of CAD diagrams.

Chapter 5 evaluates and discusses the results, investigating the proposed system's strengths and weaknesses. It also investigates model accuracy, execution time, and industrial scalability potential, supported by qualitative and quantitative evidence.

Chapter 6 summarizes the thesis by summarizing essential results, contributions to warehouse automation and management, and prospective research directions. It discusses the potential to refine how models perform across different conditions, increase the variety of datasets, and integrate the pipeline into real-world warehouse automation systems deployments.

This systematic method guarantees a rational and step-by-step introduction of the problem, the solution provided by AI, and its implementation in the context of warehouse digitalization.

2 Background

This chapter provides the theoretical and technical foundations necessary to understand the automated creation of the SQL commands from warehouse CAD layouts. The chapter discusses relational databases in the context of WMS, the importance of CAD plans, and the limitations of manual interpretation. It also examines computer vision methods for object detection in structural diagrams, the You Only Look Once (YOLO) family, and discusses labeling tools and data formats. The chapter concludes with a review of related studies that highlights gaps in the existing literature and positions this work in the context of intelligent warehouse automation.

2.1 Relational Databases in Warehouse Management System

2.1.1 Role of Databases in Warehouse Management System

Relational databases are one of the fundamental components of WMS. These systems enable the storage, management, and querying of operational and spatial data in a structured manner. This relational positional data of the warehouse is used to create a 3D rendering of warehouses and helps with the internal navigation. Information on warehouse components such as products, shelves, zones, and passageways is organized through interconnected tables. Especially in large-scale warehouses, the accurate representation of the relationships between these components is critical for operational efficiency (Gu et al., 2010).

WMS databases are typically built on a normalized relational schema, defining entities and their logical relationships. Tables such as products, racks, warehouse zones, and driveways represent warehouses' physical layout and structure. Attributes such as rack identifiers, zone classifications, spatial coordinates, and storage capacities encode the spatial and categorical properties of this structure. This data model supports spatial analysis and logistics planning, such as querying product locations or calculating zone capacity and internal movements (Van den Berg & Zijm, 1999).

2.1.2 Limitations of Manual Data Entry from Warehouse Blueprints

In traditional warehouse operations, the transfer of spatial and categorical information from text-based documents or engineering drawings, such as CAD layout, into a database structure is often performed manually during the facilities planning phase. Domain experts define components such as racks, zones, and aisles; these entities are then mapped in the form of entity-relationship (ER) models or structured tabular models, together with logistics engineers and database designers (Gu et al., 2010).

This expert-based approach, while still widely used, has significant limitations in the manual process of extracting spatial data from CAD diagrams and transferring it to existing schematics. Text-based reports fail to capture the geometric and spatial context of CAD plans, leading to incompatibilities between the digital database and the physical warehouse (Pujawan et al., 2017). This requires constant communication with facility designers, slows down the process, and increases the risk of human error.

Additionally, identifying structural elements such as stacks of shelves or aisles and transferring them to database tables as valid rows or foreign key relationships is extremely time-consuming and error-prone. Even a small change to the warehouse layout can require recalculations, re-entering existing data, creating significant inefficiencies in environments that require flexibility (Khan et al., 2022).

While ER diagrams can describe logical relationships, such as which rack belongs to which region, they fail to represent operationally critical spatial details, such as aisle width or the distance between racks (Gu et al., 2010). Such gaps can lead to information loss in processes such as route optimization or hazard

planning.

Given these limitations, today's WMS platforms need systems that can automatically embed geometric and semantic information from CAD diagrams into database structures. This transformation paves the way for more scalable, accurate, and adaptable data management processes (Pujawan et al., 2017). In order to build systems that can automatically extract and encode such spatial and semantic information, a deeper understanding of CAD layouts is required.

2.2 CAD Layouts in Warehousing

2.2.1 Functional Role of CAD in Warehouses

For WMS to work in harmony with the physical infrastructure, spatially accurate modeled data is needed to optimize space utilization and product mobility. At this point, CAD drawings create a basic map of warehouse areas by precisely defining the locations of elements such as racks, driveways, loading areas, and structural elements in a digital environment. According to Bhandari and Manandhar (2023), CAD diagrams include not only geometric layout but also semantic data that is critical for operational decisions. For example, labeling an area as "cold storage" reflects not only its physical location but also the knowledge that temperature-controlled products must be stored there.

Each warehouse has a unique design depending on the sector it operates in and the company's needs. However, the common components in most CAD plans are loading and unloading areas, transportation paths (vehicle and personnel passages), product shelves, corridors, and structural details such as walls or columns. As Gu et al. (2010) also stated, each of these components plays a decisive role in processes such as inventory management, placement strategies, and space optimization.

CAD based diagrams also provide scalable and adaptable digital models. This allows warehouse managers to determine, for example, which shelf to place a product on most efficiently or to plan suitable routes for automated guided vehicles (AGVs). The digitalization of the physical layout, combined with real-time decision support systems, enables warehouse operations to gain speed, accuracy, and efficiency (Altarazi & Ammouri, 2018).

In addition, CAD diagrams allow different scenarios to be tested with simulations before warehouse setup. This way, bottlenecks, overlaps in transportation routes, or unnecessary space usage can be detected and corrected in advance. According to Alicke et al. (2017), the integration of CAD based models with WMS is a critical requirement for integrated management of material handling, storage, and retrieval processes in large-scale distribution centers.

In summary, CAD diagrams are not just technical drawings but also dynamic strategy tools that enhance warehouse efficiency, flexibility, and scenario planning.

For a visual example of a CAD drawing used in warehouse layouts, see Figure 3.

2.2.2 From Design Tool to Automation Medium

CAD was first developed to digitize visual drawing processes in the fields of architecture and engineering (Encarnacao et al., 2012). However, today, the function of CAD drawings has gone much further than just creating technical drafts; they have become data sources that feed automation systems, especially in the context of warehouse digitalization.

As previously explained in Section 2.2.1, CAD models do not only describe physical layouts, but also

contain operational semantics. These multi-layered structures can be processed programmatically and converted into structured data formats. Converting CAD data into SQL-compatible formats enables direct integration with WMS decision support systems (Khan et al., 2022).

In addition, this functionality of CAD coincides with the general trends in industrial digital transformation. As Ivanov and Dolgui (2021) emphasizes, digital twin systems, cyber-physical infrastructures, and smart logistics processes require high-resolution and semantically rich spatial data. At this point, CAD becomes an interface that acts as a bridge between the physical warehouse and digital intelligence, enabling automated decision-making, real-time monitoring, and adaptive control strategies.

In short, CAD is no longer just a design output; it is a fundamental data interface that provides integration and interpretable structures between automation-oriented systems.

2.2.3 From Manual CAD Interpretation to AI-Driven Automation

Manually interpreting CAD diagrams and transferring them to database tables poses serious limitations in terms of scalability, accuracy, and efficiency in today's large-scale warehouse environments. As mentioned in Section 2.1.2, traditional approaches generally rely on experts defining structural and semantic details and transforming them into a relational model. However, considering the complexity and diversity of warehouse CAD plans, it is clear that these methods are no longer sufficient.

CAD diagrams vary greatly across industries and designers in terms of layer structure, symbol usage, and description formats (Howard & Björk, 2007). This diversity complicates the interpretation process and increases the dependence on experts. In addition, when CAD files are converted to raster formats (such as PDF, PNG), the vector data defining the relationships between objects is permanently lost (Yang et al., 2024). This makes it difficult to understand the structural integrity of the diagram from both a human and a machine perspective.

Correctly interpreting spatial meanings in CAD plans, for example, the distance between a shelf and aisle or the location of security areas, requires domain knowledge. Manually transferring these relationships to the database structure (e.g., with foreign key relationships) increases the risk of errors and becomes unsustainable for variable warehouse structures (Khan et al., 2022).

According to CL, manually evaluating a typical CAD drawing and inserting the relevant information into a database can take up to 50 man-hours. Moreover, any changes to the design, such as updates in the positions of racks or driveways, often require repeating the entire process, adding significant additional time and effort. This highlights the need for an automated solution that can adapt quickly to design changes while ensuring accuracy.

In overcoming these challenges, AI, especially deep learning-based CV models, offers a powerful alternative. Unlike rule-based systems, CV algorithms can learn structural patterns from labeled data and generalize to different CAD diagrams. Architectures such as Mask R-CNN and YOLOv9 can detect CAD components with high accuracy without requiring preprocessing (He et al., 2017). These models can learn not only geometric features but also semantic relationships between components.

In addition, AI systems pave the way for advanced applications such as digital twins, IoT infrastructures, and predictive maintenance. By converting CAD-derived data into structured formats, scenarios such as autonomous vehicle routing, energy efficiency, and security monitoring can be supported (Alicke et al., 2017).

As a result, the transition from manual processes in CAD interpretation to AI-supported automation is a strategic necessity not only in terms of efficiency but also in terms of establishing smart, flexible, and digitally ready warehouse systems.

2.3 Computer Vision for Diagram Understanding

CV is a subfield of AI that focuses on the perception and interpretation of visual data. It enables machines to perform tasks based on visual inputs by automating tasks such as image classification, segmentation, scene analysis, and object detection (Gonzalez & Woods, 2018; Szeliski, 2010). The analysis of structural images used in warehouse management, in particular, requires solutions beyond traditional computer vision methods. Such images have low visual complexity but high geometric regularity compared to natural images. These diagrams also include abstract visual elements such as symbols, lines, and annotations, which were previously discussed as part of their semantic layer (see Section 2.2.1 and Section 2.2.2).

Therefore, CV models to be used in CAD-based images need to be customized by taking these structural features into account. In this context, object detection stands out as a fundamental task. Object detection is a CV problem that aims to simultaneously determine both the location (e.g., with bounding boxes) and the class of objects in an image. Extracting meaningful and structured information from CAD diagrams depends greatly on the success of this fundamental task. This section discusses the technical evolution of object detection in structured images, the differences between CAD and natural images, and the detection challenges specific to CAD, respectively.

2.3.1 Object Detection in Structured Images

Structured images are types of visual representations that contain predefined symbols and geometric patterns. CAD diagrams also fall into this category and are of great importance for object detection, especially in areas such as warehouse automation.

Early object detection approaches relied on human-defined features such as edges, textures, or shapes. For example, the Haar cascade method is effective in recognizing specific patterns, while methods such as (Viola & Jones, 2001) and HOG-SVM aimed to classify based on edge information(Dalal & Triggs, 2005). However, these methods were not robust against changes in size, angle, or environment.

By overcoming these limitations, deep learning-based approaches have revolutionized the field of computer vision. In particular,(Convolutional Neural Network (CNN)) has enabled self-learning of images, providing higher accuracy in object detection. R-CNN and its successor models Fast R-CNN and Faster R-CNN have increased accuracy rates by optimizing the region proposal and classification stages (Girshick, 2015; Girshick et al., 2014; Ren et al., 2015). However, due to their multi-stage structures, these models have long processing times and are limited for real-time applications.

Therefore, single-stage detectors have been developed. Models such as YOLO and SSD (Single Shot Detector) have balanced speed and accuracy by simultaneously estimating the location and class of objects (W. Liu et al., 2016; Redmon et al., 2016). The YOLOv8, YOLOv9, YOLOv10, and YOLOv11 versions developed in recent years have been optimized for small object detection, color and symmetry sensitivity, data augmentation, and lightweight model structures, especially for structured diagrams such as CAD (Ultralytics, 2023, 2024a, 2024b; C. Wang et al., 2024).

These models can effectively learn the features previously described about the semantic content and regular structure of CAD diagrams (see Section 2.2.1). This allows elements such as shelves, roads, and symbols to be recognized more consistently with less variation. Furthermore, these detection outputs can be structured in a way that enables their integration into relational databases, supporting the automatic CAD-to-database conversion process discussed in Section 2.2.3.

2.3.2 Differences Between CAD and Natural Images

The success of CV techniques largely depends on the type of visual data being studied. In particular, CAD diagrams have quite different structural properties compared to the natural images that computer vision models are accustomed to. These differences reveal why classical visual recognition approaches are inadequate in CAD environments and why CAD-specific strategies are needed (W. Zhang et al., 2023).

Unlike CAD drawings, natural images—such as real-world photographs—present complex scenes with irregular, dense textures and color gradients. In these scenes, CV systems use elements such as edges, texture, light-shadow relationships, and color differences to detect objects (Hancock et al., 1992). As previously discussed in Section 2.2.2, CAD diagrams are typically composed of geometric shapes defined by lines, arcs, polygons, and symbolic labels. In environments with such low visual variety, traditional CV models can struggle, especially in object classification and segmentation tasks (Yang et al., 2024).

In addition, CAD diagrams are texturally simple but spatially complex structures. For example, although in a warehouse drawing, racks, aisles, or walls are modeled with their unique geometry, there is no color or texture distinction between these objects. However, many computer vision algorithms learn by relying on textural cues in the training data. Therefore, specialized models with greater spatial context and shape sensitivity are required to be effective on CAD images (Yang et al., 2024; W. Zhang et al., 2023).

Another difference between natural images and CAD drawings is that, as previously mentioned when discussing CAD diagrams for Automation and Control (see 2.2.2), CAD drawings contain meaningful textual and symbolic annotations to be explanatory to the viewer. These can be text, such as labels for the objects they contain, distance measurements, size measurements, and area names. However, most of these annotations are displayed in small fonts and are located between tightly packed objects. This forces computer vision systems to perform special tasks such as small object detection and symbol separation (W. Liu et al., 2024).

Although CAD drawings are usually created in vector formats, in practice they are often converted to raster format for purposes such as printing, archiving and visual analysis (Intwala, 2020). As we mentioned (see 2.2.3), this transformation causes the loss of important metadata such as layer information and structural relationships between objects in the drawing. Also, with natural images, the camera properties like focal length, aperture can be used to evaluate the depth of the image, which will help the model to train better. As a result, computer vision systems are forced to rely solely on pixel-based visual data. This may be insufficient for accurate object detection and classification, especially in complex layouts. Therefore, to effectively process CAD diagrams, not only advanced detection algorithms but also preprocessing techniques (e.g., thresholding, data augmentation) specifically adapted for such structural images are needed (Yang et al., 2024).

All these structural differences show that classical approaches developed in the field of computer vision cannot be directly applied to CAD. Therefore, object detection strategies developed for CAD-specific images should be designed by taking into account the characteristics of these artificial environments and adopting a different training paradigm than classical natural image datasets (W. Zhang et al., 2023).

2.3.3 Object Detection Challenges in CAD

The difficulties that arise when applying object detection models to CAD diagrams are distinct from those that arise in the analysis of natural images. The small object detection difficulty is one of the main challenges. Objects like racks or origin points sometimes take up extremely few pixels in CAD layouts in comparison to the size of the entire image. Small objects typically lose representational information during deep network processing, particularly after numerous convolutional and pooling layers, which lowers identification accuracy (Tong & Wu, 2024).

Another challenge of the task is the cluttered appearance of warehouse scene layouts. Contrary to real-world scenes in which elements may be widely separated, CAD drawings are characterized by having hundreds of features that are very close together. The detection and distinction of such objects necessitates high-resolution inputs and detection models that can perform fine-grained localization. Many methods have been devised to address dense detection tasks, such as multi-scale training, optimization of the anchor boxes, and improved Feature Pyramid Network (FPN) (T.-Y. Lin et al., 2017).

Differences in CAD drawing standards within sectors also lead to discrepancies in labeling norms, line weights, and symbol representations. If CAD diagrams from one warehouse differ greatly from those from another, a model trained on the former may find it difficult to generalize to the latter. To overcome this difficulty, domain adaptation strategies, significant data augmentation, and cautious dataset diversification are frequently needed (Buslaev et al., 2020).

Another challenge is the introduction of noise and artifacts during CAD rasterization. As previously discussed in Section 2.3.2, the conversion of vector-based CAD files into raster pictures (such as PNG or JPEG) may result in slight pixelation effects, line breaks, and distortions. Detection models may become confused by these artifacts, particularly if the training data is not adequately representative of these fluctuations.

Finally, spatial context is important when reading CAD diagrams. Identifying objects is not enough; one needs to understand the relative positioning of objects, such as a rack lined up along a driveway, for proper semantic interpretation. Methods that merge object detection and spatial reasoning, graph-based modeling, or rule-based postprocessing have been proposed for solving this issue (Zheng et al., 2022).

2.4 Annotation Tools and Data Formats in Computer Vision

2.4.1 Importance of Annotated CAD Warehouse Images

Deep learning models for object detection need large, varied, precisely annotated datasets. Regarding CAD drawings, the structured character of the images, defined by geometric primitives, sparse labeling, and dense object layouts, demands even more accuracy in annotations (Heidari & Iosifidis, 2024).

CAD modeled warehouse layouts often require labeling of critical components such as racks, driveways, and zone boundaries. High-quality labeled datasets enable the model to learn the visual and spatial differences between these components, enabling the model to perform consistently across a variety of warehouse layouts with different design styles and international drawing standards (Zhao et al., 2020).

Beyond visual learning, these annotations also play a critical role in measurement, scaling, and schema-based data extraction. In recent studies, point-based annotation strategies have been preferred over area-based ones (Zheng et al., 2022). This approach allows for more accurate modeling of object centers and structural reference points.

2.4.2 Dataset Annotation Tool: CVAT

Producing annotated datasets for technical diagrams requires specialized software offering precision and scale. CVAT, developed by Intel (Sekachev et al., 2020), is widely used in such tasks due to its flexibility and support for both object-level and keypoint annotations.

CVAT supports various annotation types, such as bounding boxes, polygons, polylines, and keypoints, in addition to precision-enhancing features like zooming, snapping, and grid overlays. These functionalities are especially useful when working with CAD diagrams that contain high-density geometric features and

require pixel-level precision (Sekachev et al., 2020).

In structured environments, annotation strategies generally include point-based annotations that include information such as reference points of objects (e.g., starting points, compartment boundaries), as well as detections made with bounding boxes. Such structured annotations enable the model to better learn both spatial context and semantic relationships (Heidari & Iosifidis, 2024).

2.4.3 Annotation Format: YOLO

Annotation formats determine how labeled data is organized for model training. The YOLO annotation format, first proposed by Redmon et al. (2016) and extended in subsequent versions (Ultralytics, 2023), encodes each object instance in a compact text line format containing its class label and normalized bounding box coordinates.

This format provides an efficient structure, especially for real-time object detection applications. The latest versions of YOLO support not only bounding boxes but also keypoint-based representation formats. This helps the model learn semantic references in structural images such as CAD more effectively (Ultralytics, 2023).

2.5 Data Preprocessing and Augmentation in CAD Contexts

2.5.1 Preprocessing Techniques for CAD Drawings

Technical CAD drawings can often be inconsistent due to resolution differences, background colors, and contrast issues caused by outputs from different software or scanning processes. Such visual artifacts negatively affect the success of computer vision models, especially those that attempt to detect small and sparse objects. Therefore, preprocessing techniques such as adaptive thresholding and binarization are vital for improving geometric clarity in CAD images.

Adaptive thresholding determines the threshold value specifically for each region in the image and compensates for lighting irregularities better than fixed thresholding (Gonzalez & Woods, 2018). Binarization, on the other hand, increases edge detection success by reducing color clutter and allows the model to focus only on structural information (R. Zhang et al., 2018).

2.5.2 Visual Cue Encoding in Annotated Data

In visual environments with sparse labeling, such as CAD, components are often represented by a small number of pixel changes. Therefore, visual encoding of semantic cues facilitates the model to learn both structural locations and meanings. The colored dots used in annotations provide the model with the opportunity to learn class information as well as positional information.

This approach overlaps with the idea of visual prior, where semantic meaning is encoded in a visual form and incorporated into the model's learning process. This can improve model accuracy, especially in low-texture drawings such as engineering schematics and floor plans.

2.5.3 Data Augmentation in Sparse Annotation Domains

One of the biggest challenges when developing deep learning models for engineering drawings is the lack of sufficient labeled data. Since CAD images are usually created for human-to-human communication, the labeled data required for machine learning is often limited. Therefore, data augmentation is an effective solution to both expand the dataset and increase the generalization ability of the model.

Augmentation operations performed in sparsely labeled environments, such as CAD, need to preserve geometric integrity. Therefore, operations such as small-angle rotations, brightness/contrast adjustments, and horizontal reflection are preferred. Libraries such as Albumentations are often preferred for such augmentations because they offer flexibility and performance (Buslaev et al., 2020).

In addition, thanks to augmentation, the model learns to recognize the same structure under different views, preventing overfitting (Shorten & Khoshgoftaar, 2019).

2.6 Object Detection Models

2.6.1 YOLO Family and Advances (YOLOv8–YOLOv11)

In the last decade, significant progress has been made in object detection. The YOLO family of object detection models has played a significant role in the field of real-time visual recognition. Unlike two-stage detectors such as Faster R-CNN, which first generate region proposals and then perform classification (Ren et al., 2015), the YOLO architecture formulates object detection as a single-stage regression task. This allows it to simultaneously predict bounding boxes and class probabilities directly from raw input images in one forward pass, which will significantly improve speed without major sacrifices in accuracy (Redmon et al., 2016).

YOLOv8 marked a major leap forward with its anchor-free design and decoupled detection head. This is also a model with improved accuracy and training stability. Its modular structure supports object detection, segmentation, and pose estimation, making it a versatile choice for real-time applications like analyzing CAD layouts (Ultralytics, 2023).

YOLOv9 tackles the common problem of information loss in deep networks by introducing Programmable Gradient Information (PGI) and reversible functions. This ensures a better gradient flow and data retention. With its efficient GELAN architecture, YOLOv9 delivers high accuracy and strong performance, especially in lightweight models, while maintaining real-time speed (C. Wang et al., 2024).

YOLOv10 introduces a refined architecture that removes the need for Non-Maximum Suppression (NMS), enabling faster inference. With optimizations like lightweight heads and attention modules, it balances high accuracy along with efficient performance across various detection tasks (A. Wang et al., 2024).

The latest installment in the series, YOLOv11, is a fast and efficient object detection model with improved accuracy and fewer parameters compared to earlier versions. It supports a wide range of computer vision tasks and works well across different environments, from edge devices to cloud platforms (Khanam & Hussain, 2024).

To faithfully illustrate the performance leap among various YOLO versions, Figure 1 graphs YOLOv8 to YOLOv11 against other state-of-the-art detectors on detection accuracy and inference latency. The x-axis represents latency (in milliseconds per image) as measured using TensorRT10 on an NVIDIA T4 GPU at FP16 precision, and the y-axis represents COCO (mean Average Precision (mAP) 50–95)—a standard metric of average object detection accuracy across a range of Intersection over Union (IoU) thresholds from 0.50 to 0.95.

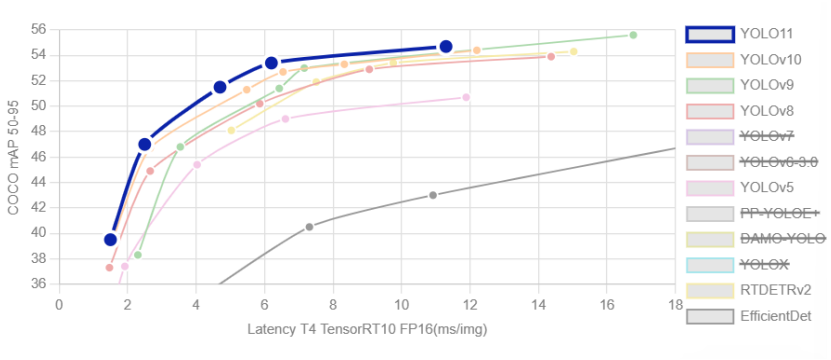


Figure 1. Comparison of YOLO Family Models and Competing Detectors on Accuracy vs. Latency (COCO mAP 50–95, T4 TensorRT10 FP16) *Adapted from (Ultralytics, 2023). Note: Although our thesis does not use the COCO dataset, this benchmark is included for theoretical comparison and performance context.*

The plot demonstrates that YOLOv8 achieves the highest inference speed with a slight trade-off in accuracy. YOLOv9 and YOLOv10 present a more balanced trade-off between accuracy and latency. Although small object detection performance is not directly shown in the figure, prior studies report that YOLOv9 and YOLOv10 improve detection of small and densely packed objects through architectural enhancements such as PGI and CDA strategies (A. Wang et al., 2024; C. Wang et al., 2024). The most significant overall gain is observed in YOLOv11, which reaches the highest accuracy (55 mAP) while maintaining a low latency (6–8 ms). This makes YOLOv11 especially suitable for time-critical tasks such as parsing structured CAD layouts.

The consistently stronger performance of YOLOv11, as identified by the stark blue line, serves to reaffirm its dominance over earlier versions of YOLO and other detection models, including EfficientDet, RT-DETR, and YOLOv5.

The evolution process from YOLOv8 to YOLOv11 represents a special milestone in the development of more complex, efficient, and generalizable object detection models. These advances are especially important in real-world applications, such as the automatic conversion of CAD diagrams into structured database schemas. In such scenarios, not only detection accuracy but also inference speed and computational efficiency are critical factors determining system performance. Therefore, the models used need to be evaluated not only in terms of accuracy but also in terms of computational complexity. Floating Point Operations (FLOPs) is a theoretical measure of the number of arithmetic operations a neural network can perform during a single forward pass (inference). This value, usually expressed in billions (GFLOPs), is used to estimate the computational load of a model. In particular, for object detection tasks that require real-time processing of high-resolution warehouse CAD diagrams, models with lower Floating Point Operations (FLOPs) are preferred because they offer faster inference times. This makes them suitable for low-latency, resource-constrained applications (A. Wang et al., 2024; C. Wang et al., 2024).

To better understand how these models are trained and fine-tuned for specific applications, it is useful to briefly introduce some of the key training parameters commonly used in YOLO based detection systems:

- Box: Weight applied to the bounding box regression loss. A higher value emphasizes accurate object localization, following standard YOLO settings (Redmon et al., 2016).
- Cls: Weight applied to the classification loss (VFL or BCE), impacting how much the model focuses on correct class prediction (T. Y. Lin et al., 2017; H. Zhang et al., 2021).
- HSV H, S, V: Hyperparameters for hue, saturation, and brightness jittering during data augmentation. Such augmentations help generalize to lighting and color variations and are commonly used in large-scale object detection systems (Buslaev et al., 2020).

- Mosaic: Proportion of mosaic augmentation (combining four images into one), improving the model's ability to detect small and occluded objects. Mosaic was popularized in YOLOv4 and is inherited in subsequent versions. It also helps reduce overfitting by exposing the model to diverse object combinations (X. Zhang et al., 2023).
- Scale: Defines the random scaling range applied to training images, contributing to scale invariance. Typical values around $[0.5, 1.5]$ are used to augment object sizes and improve detection across variable object dimensions (Buslaev et al., 2020).
- VFL: Indicates whether VFL is used for classification instead of BCE. VFL is designed to handle class imbalance and crowded scenes better, as shown in (H. Zhang et al., 2021).

2.6.2 Evaluation Metrics

There are three widely used metrics to evaluate the effectiveness of object detection models, especially in structured data domains such as CAD diagrams: Precision, Recall, and mAP. These metrics are based on information retrieval theory and have become standard in CV benchmarks such as Pascal VOC and COCO (T.-Y. Lin et al., 2014). Precision measures how many of the samples predicted as positive by the model are correct and is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

where TP is the number of true positives and FP is the number of false positives. Higher precision means that the model produces fewer false alarms.

Recall (Sensitivity) measures the proportion of actual positives correctly identified by the model:

$$\text{Recall} = \frac{TP}{TP + FN}$$

Where FN represents the number of instances the model failed to detect. High recall means fewer missed detections.

mean Average Precision (mAP@0.5:0.95) is a widely used composite metric for object detection tasks. It aggregates the average precision (AP) scores over multiple IoU thresholds ranging from 0.50 to 0.95 in steps of 0.05:

$$\text{mAP} = \frac{1}{10} \sum_{i=0.5}^{0.95} \text{AP}_{\text{IoU}=i}$$

Here, $\text{AP}_{\text{IoU}=i}$ denotes the area under the precision-recall curve at a specific IoU threshold. The precision–recall curve shows the relationship between *precision* and *recall* at different confidence score thresholds. ROC curve plots the true positive rate against the false positive rate which evaluates a classifier's overall ability to distinguish between classes. Unlike the ROC curve, the precision–recall curve is considered more informative, especially in scenarios where the class distribution is unbalanced, such as object detection. mAP calculation is widely used to evaluate both localization and classification performance by averaging AP values across multiple IoU thresholds.

These metrics are of critical importance, especially in structure-dense CAD environments, as over-detection or under-detection can cause structural data inconsistencies. Using Precision, Recall, and mAP together,

the accuracy and integrity of models can be evaluated in a balanced manner.

2.7 Spatial Reasoning and Postprocessing in CAD

Although object detection models such as YOLOv8–YOLOv11 can successfully detect components in warehouse CAD diagrams, the outputs of these models usually bounding boxes with class labels or center coordinates cannot be directly used in real-world applications. These outputs must undergo an additional postprocessing step to be semantically interpreted and integrated into downstream applications. This step includes operations such as inferring spatial relationships between objects, scaling, and position verification.

In warehouse automation systems, detecting objects such as racks or driveways is insufficient. To transfer these components to the relational database schema, their locations, distances to each other, and alignment must be determined with high spatial accuracy. Therefore, operations such as converting the detected coordinates from pixel to metric system, calculating distances between objects, and verifying alignment angles are critical to closing the gap between visual perception and structural data generation (Zheng et al., 2022).

Various geometric strategies exist for measuring inter-object relationships and aligning detected elements to physical layouts. For example, pixel-based measurements can be transformed into metric values by applying scaling coefficients derived from CAD specifications. These transformations are essential for applications such as layout optimization, zone assignment, and path planning in logistics environments (Tong & Wu, 2024).

Additionally, in industrial environments where high precision is required, user-assisted (manual-in-the-loop) verification mechanisms are often used to increase the accuracy of model outputs. For example, OpenCV-based click-based validators allow engineers to compare model outputs with ground truth values by clicking on predicted locations in the image. Such interactive verifications both increase the reliability of model outputs and enable fine-tuning in environments where even minor positional errors can cause significant operational problems (Rosebrock, 2015).

Integrating post-processing techniques into the CAD-to-database transition enhances the semantic interpretability of computer vision outputs. By incorporating spatial reasoning principles into the system design, storage components can be placed with spatial awareness, while ensuring that records in the database remain true to the actual physical structure. It is emphasized in the literature that object detection systems without such spatial reasoning steps have limited utility downstream (e.g., in SQL generation) (Zheng et al., 2022).

2.8 Experiment Tracking and Training Monitoring Tools

Experiment tracking platforms have become indispensable in modern AI applications to ensure a reliable and repeatable training process in deep learning processes. Comet.ml, one of the standard tools in this field, provides real-time monitoring of model training, hyperparameter tuning, and performance comparisons between multiple experiments. It keeps detailed records of metrics such as loss, accuracy, precision, and recall, and visualizes outputs such as training curves, confusion matrix, and model checkpoints via a central control panel. Especially when training object recognition models with custom datasets, Comet.ml facilitates systematic evaluation of model behavior and collaborative model development processes within the team. In this study, the training and fine-tuning processes of YOLO-based models developed for CAD-to-SQL automation were monitored with Comet.ml; thus, the most appropriate training configurations were determined and the generalization performance of the model was monitored (Comet ML, 2024).

2.9 Related Work

2.9.1 CAD-to-Database Conversion Approaches

Transforming CAD plans into structured database representations has been addressed in many fields such as Geographic Information Systems (GIS), engineering, CAD, and construction. Conversion of raster-based CAD images, which are frequently encountered in logistics, into meaningful and spatially structured digital formats is a significant challenge.

Niu et al. (2015) has developed a declarative feature recognition system that translates geometric descriptions into SQL queries to enable scalable identification of structural patterns within CAD models. Although this method has been implemented for vector-based CAD data, it demonstrates the potential of database-driven inference systems in information mining.

It is seen that there is an increasing need for systems that can bridge the gap between the interpretation of raster-based CAD diagrams and structural database modeling processes. Especially in applications such as warehouse automation, data-driven approaches focusing on object recognition and spatial relationship inference are critical to overcoming practical challenges.

2.9.2 Object Detection in Engineering and Technical Diagrams

Recent studies have shown that traditional object recognition methods are inadequate in understanding complex engineering drawings. Bhanbhro et al. (2022) has revealed the limitations of traditional methods, especially in drawings containing dense and symbol-filled structures such as P&ID (Piping and Instrumentation Diagrams), and compared the performance of deep learning models such as YOLO and Faster R-CNN. These models offer significant advantages in identifying symbols and establishing connections correctly. Similarly, Nurminen et al. (2020) examined the applicability of YOLO-based object recognition methods, especially for extracting information from old engineering diagrams. It improved the detection of small objects by introducing a strategy of dividing large diagrams into small subsections. This approach is also an important method adopted in our thesis.

As a solution to the problem of data insufficiency, Niemistö (2024) investigated the effectiveness of few-shot learning and synthetic data augmentation techniques in the engineering context. He showed that reliable object recognition performance can be achieved even when the number of labeled data is small. These findings are of great importance for data-driven methods that work with limited annotation, especially in raster-based CAD diagrams.

When these studies are evaluated together, the need for object recognition approaches specific to engineering diagrams becomes even more evident. Automatic identification of structural components (e.g., shelves, passageways) from raster format CAD diagrams and transferring this information to the digital environment together with their spatial relationships stands out as one of the fundamental research problems in the field. Therefore, there is increasing interest in the use of advanced YOLO-based models aimed at extracting structural information from technical diagrams.

2.9.3 Vision-Based Layout Understanding in Warehouse Logistics

CV is transforming warehouse management processes in the logistics industry by enabling the automatic interpretation of spatial layouts. In a comprehensive literature review presented by Naumann et al. (2023), vision-based applications in transportation and warehouse logistics are examined under two main headings: monitoring (e.g., inventory tracking, environmental observation) and interaction (e.g., robotic

intervention, layout adjustment). The review study highlights the increasing interest in extracting spatial context from warehouse environments using deep learning models and 2D/3D visual data. Furthermore, the lack of domain-specific datasets and the difficulty of developing generalizable solutions to different warehouse layouts are identified as key research gaps.

In this context, Yin et al. (2022) presented a new machine vision-based inventory method for detecting and counting stacked products in stereoscopic warehouses. Using the STCNet model based on Swin Transformer, this method detects product surfaces from complex scenes with high accuracy. Combined with a robust counting algorithm, the system achieved high accuracy on specific datasets and proved its scalability in real warehouse environments. This approach demonstrates how layout analysis in logistics can benefit from hybrid techniques that combine both object recognition and structural reasoning.

When these studies are evaluated together, the critical role of computer vision in automating spatial understanding in warehouse environments becomes clear. As warehouse diagrams become increasingly complex and diverse, robust vision-based systems that can cope with challenges such as overlapping, stacking, and limited labeling are of great importance for the development of smart warehouse automation.

2.9.4 Visual-to-Structure Transformation for CAD Based Information Systems

The transformation of CAD diagrams into structured information systems is becoming increasingly important with the rise of digital twins and automation. One of the early works in this area was Flynn and Jain (1989), who proposed a geometric inference system that transforms 3D CAD object descriptions in IGES format into relational graphic structures. This work pioneered the idea of structural inference in the field by showing that basic geometric elements in CAD can be transformed into structural representations suitable for object recognition and database generation. However, this approach is based on the vector CAD format and requires much rule-based engineering knowledge.

One of the recent works, S. Wang et al. (2025), introduced a multimodal large language model called CAD-GPT that can generate CAD construction sequences from visual and textual inputs. This method improves spatial reasoning with 3D Modeling Mechanism, enabling accurate extraction of origins and orientations of geometric structures. Although the study focuses on CAD model generation rather than direct drawing interpretation, it is similar to the image-to-database transformation approach proposed in this thesis in that it combines natural language and visual information to produce structured geometry.

In addition, the CAD2Program method developed by X. Wang et al. (2024) encodes 2D raster CAD drawings using Vision Transformer and produces 3D parametric model definitions from these drawings. This method offers a modern approach to visual-to-structure transformation without the need for traditional vector inputs.

When these studies come together, it is understood that the need for AI supported multi-modal systems is increasing in interpreting and structuring CAD data.

2.9.5 Gaps in Current Research and Motivation for This Study

While each of the topics discussed above is important in its own right, studies that integrate these components into an end-to-end system are limited. Existing approaches are either specific to a specific industry (e.g., architecture or mechanics) or focus on extracting document-level information that ignores spatial relationships.

Furthermore, comprehensive studies on processing image-based drawings are lacking in logistics scenarios where only raster CAD outputs are available. This thesis addresses these gaps by aiming to interpret

warehouse CAD diagrams and convert spatial relationships into SQL commands using deep learning-based computer vision techniques.

No studies try to extract positional information from CAD drawings to create a structural output. Also, the approaches mentioned here try to read text from the image to get information. At the same time, in our case, there are many texts other than what we need, and the drawings are from different countries and in various languages, making the existing systems unusable.

3 Method and implementation

3.1 Overview of the Approach

This thesis presents a deep learning approach for automatically extracting structural information from warehouse CAD layouts, using CV techniques within AI engineering. The core objective is to design and implement a robust AI pipeline that transforms complex, large-scale technical diagrams into structured, queryable representations. An overview of the complete methodology pipeline—from CAD preprocessing to SQL generation—is illustrated in Figure 2.

The process began with preprocessing the CL acquired warehouse CAD plans. Due to the dense composition and massive sizes of such technical drawings, cropping was the preferred method for dividing up small regions of interest to allow for more data volume, along with greater model concentration on features such as driveways and racks. To achieve generalization, CAD drawings were utilized from different countries and clients. This introduced variability in symbol use and drawing styles, thus increasing the robustness of the learning process.

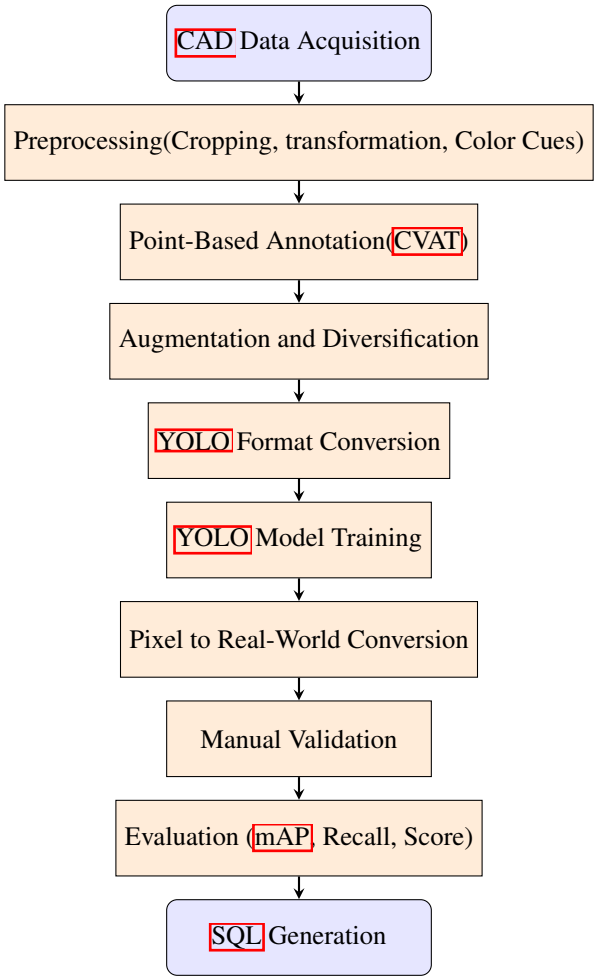


Figure 2. CAD to SQL pipeline

The annotated data was generated by a point-based labeling method via the CVAT tool, in which significant features such as racks, driveways, and source points were marked as single points instead of bounding boxes. Since a huge warehouse has multiple rooms, the images are cropped accordingly, and an origin is

marked, a reference for the other elements' position. The distance to various objects is considered from this origin. This will be mostly the top left side of the warehouse or the room.

This choice was driven by the geometric nature of warehouse layouts and the need for precise localization, especially regarding future measurements and SQL logic. The annotations were then exported and can be converted into different formats to support different AI models.

The dataset was extensively augmented using the Albumentations library (Buslaev et al., 2020) to ensure sufficient data diversity and model resilience. This included controlled rotations, flips, brightness, and contrast adjustments to simulate variations. The augmented data was divided into training, validation, and test sets using stratified sampling to avoid data leakage and class imbalance. Along with this, some images are preserved without augmentation, for manual testing. These images are never used to train or in the validation process, making them fresh and new images to model.

Diverse Yolo detection models(YOLOv8-11) were trained concurrently to analyse the detection quality. A Varifocal Loss and a greater box loss weight were used to fine-tune the YOLO models to enhance the detection of small and overlapping features, which are more characteristic of CAD layouts (H. Zhang et al., 2021). Training was done on GPUs, and performance was monitored and validated with Comet.ml (Comet ML, 2024).

Following inference, the model outputs were post-processed to calculate pixel-based distances between detected features and the origin. These distances were translated into real-world units based on known CAD scale factors. A manual OpenCV-built validation tool was used to cross-validate the predictions. The final model is identified based on different evaluation metrics used.

In summary, the proposed AI-based method exhibits the potential to transform unstructured computer-aided design information into structured knowledge representations. The use of multiple models, along with careful preprocessing and training steps, provides a flexible and scalable approach to understanding visual data. This method can be directly applied to engineering drawings where the measurements and positions need to be extracted structurally.

3.2 Dataset Preparation

3.2.1 CAD Image Collection and Cropping

The basis of building the AI pipeline was the acquisition of CAD layouts of warehouse plans. The first data was in the form of large technical layouts of CL's warehouses, as in Figure 3. The initial layouts contained detailed warehouse information, including rack positions, driveways, zone markings, and structural aspects.

However, due to the size and high element density of the diagrams, it was not possible to employ full-scale layouts as a training model. Thus, a cropping technique was employed to cut down each big CAD layout into smaller but still useful portions. The cropping technique had two primary functions: it raised the quantity of training samples and enabled the models to focus on localized areas of interest with meaningful features. There are some drawings in document format, and others were DWG (Auto CAD format). These images were cropped programmatically using a command-line tool called pdftoppm (Glyph & Cog, 2025) by providing crop positions and dimensions.

Every cropped photo preserved the essential visual characteristics of the original design while, at the same time, lowering the computational complexity of high-dimensional images in the process of training models.

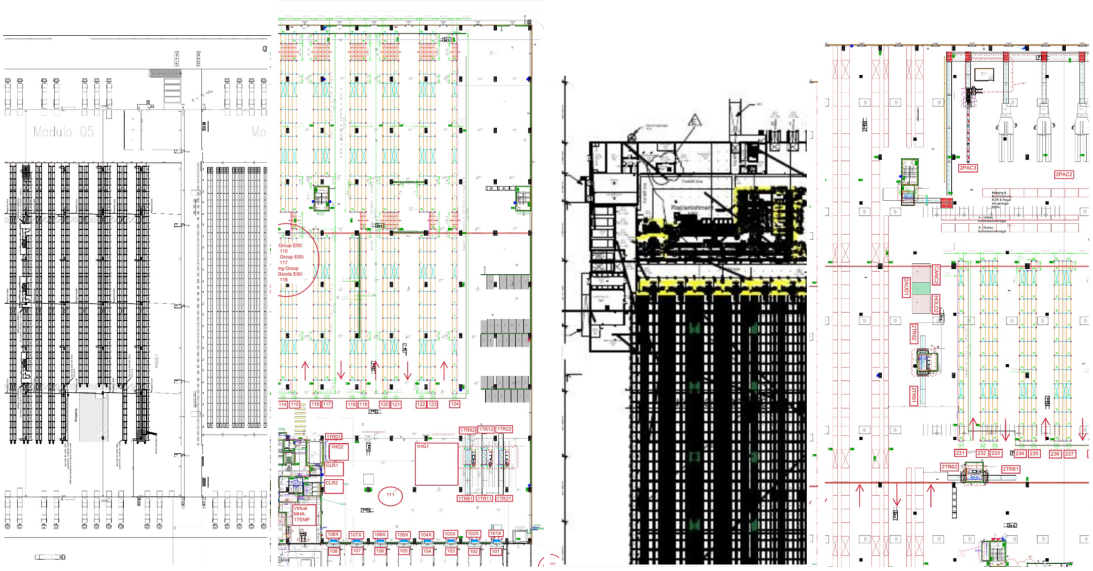


Figure 3. Sample warehouse CAD drawings cropped into different images

3.2.2 Dataset Diversification

The initial cropped perspectives from CAD drawings of a single warehouse layout were a useful starting point, but they were insufficient to train a model that would work well with different drafting standards and warehouse configurations. In terms of spatial alignment, resolution quality, line type, and symbol convention, CAD drawings can also differ greatly between organizations and geographical locations.

To solve this problem, more warehouse CAD layouts were requested from CL. The expanded dataset included several warehouse layouts from different countries and customer organizations; each one showed differences in diagrammatic conventions because CAD diagrams for warehouses vary from country to country and organization to organization, as in Figure 3.

Exposure to a wider range of layout styles and drawing conventions helped the system handle diagrams it hadn't seen before. This made the pipeline flexible enough to work with warehouse CAD drawings from different industries, without needing a lot of retraining.

3.3 Data Annotation

Precise annotation of data is an integral part of developing a reliable detection system, particularly for technical drawings such as warehouse CAD layouts. Due to the complicated nature, precision, and large quantity of objects displayed in warehouse CAD drawings, a rigorous process was required to guarantee the correctness and meaningfulness of the annotations to the subsequent machine learning models.

3.3.1 Annotation Strategy

Initially, bounding box annotations were considered to be used for labeling the locations of racks and driveways. Later, it became apparent that this wasn't enough to capture the special needs in parsing warehouse layouts. Representing whole racks or driveways using rectangular boxes did not adequately represent the fine-grained relationships needed for operations like section identification, finding a starting

point for racks, and measuring distances.

Therefore, a point-based annotation method was adopted using the Computer Vision Annotation Tool (CVAT) (Sekachev et al., 2020). Each CAD drawing of a warehouse was marked with three types of points: the Origin (marked in blue), which serves as the spatial reference for scaling and distance calculation; Racks (marked in red), which represent starting point of a particular rack along each storage line; and Driveways (marked in yellow), which represent boundary dividers among adjacent rack sections.

The annotation convention rigidly maintained all the origins, rack sections, and driveways uniformly marked in all the CAD drawings. This was especially important considering the diversity in warehouse designs CL had to offer, from differences in rack density, orientation, and spacing.

The application of a point-based labeling system significantly eased the following operations, such as pixel distance measurement, section segmentation, and SQL generation. Since the models had to predict individual keypoints instead of larger object boundaries, this method reduced ambiguity and improved the robustness of the automated pipeline.

3.3.2 Exporting Annotations

The annotation process used CVAT to split images into training, validation, and test sets. All annotations were saved in the "Images 1.0" XML format, which made it easier to convert the data into other formats. Using CVAT helped create a smooth workflow for the entire data pipeline. The point-based annotations worked well with CAD layouts, even when the drawing styles varied. This allowed the models to generalize without needing retraining for each new layout.

3.4 Image Preprocessing

The preprocessing of CAD diagrams was required to improve data quality and usability before training detection models. CAD warehouse layouts vary in line density, background color, and noise. In the absence of preprocessing, such variability can create inconsistencies that adversely affect model performance. Therefore, some transformations were conducted on the original images to normalize their appearance, enhance object visibility, and mark annotated points so that model learning would be effective.

3.4.1 Color Transformation

The initial preprocessing step was to transform all warehouse CAD diagrams into black-and-white images. The CAD files gathered from CL differed greatly — some diagrams were completely colored, whereas others had light or noisy backgrounds. Colored diagrams, although more informative to human viewers, could introduce misleading color gradients for CV models (R. Zhang et al., 2018), where the importance of controlled augmentations and visual consistency is emphasized.

To address this problem, an approach of adaptive thresholding was adopted (Gonzalez & Woods, 2018). Unlike global thresholding, adaptive thresholding determines the threshold for separating smaller sections of the image, thus proving to be very effective for diagrams with non-uniform lighting conditions or backgrounds of varying shades. Therefore, to improve the readability and contrast of the CAD drawings, adaptive thresholding techniques were applied (Gonzalez & Woods, 2018). Thus, even CAD designs with reduced visibility of lines or inhomogeneous gray backgrounds were successfully binarized, resulting in images with good contrast and sharp geometric shapes. Black-and-white conversion helped detection models concentrate solely on spatial structure without getting distracted by irrelevant visual noise.

3.4.2 Adding Visual Cues

Following the black-and-white conversion, additional visual information was cast over the preprocessed images. Specific colored dots were overlaid on the images to indicate the elements manually labeled: origin points, racks, and driveways. Specific color codings were assigned to each class to allow models to distinguish between object types not only by location but also by color signature.

To help the model distinguish between similar-looking features in the thresholded CAD images, colored markers were added during annotation. Blue markers indicated the "Origin" point, which served as the reference for calculating distances. Red markers marked the start of Racks, while yellow markers were used to identify driveways. Without these markers, thresholding would have made the diagrams appear too uniform, making it difficult for the models to tell apart different structural components. The use of color overwriting helped preserve important positional and structural cues.(Creek & Mullins, 2022).

The visible signals also provided good supervised signals during model learning, which strengthened object detection accuracy and keypoint localization accuracy (Bao et al., 2023; X. Zhang et al., 2023). Furthermore, due to the size of the points and the proper placement, neither occluded CAD features nor blocked the interpretation of the diagrams, yet added richness without hiding underlying features, leaving model interpretability intact.

3.5 Data Augmentation

To increase the training set's diversity and resilience without more CAD layouts, data augmentation was an essential step. It also simulates the variances and irregularities that models would experience in the real world during inference.

3.5.1 Augmentation Techniques

The augmentation process was designed to create synthetic variants of the original CAD diagrams while preserving their geometric integrity and semantic relationships. This prevented model overfitting and allowed the detectors trained to generalize better to new, unseen diagrams.

For this purpose, the Albumentations library (Buslaev et al., 2020) was used to create altered copies of the original images. Various photometric and geometric transformations were implemented, which consisted of random rotation, horizontal flip, and adjustments of brightness and contrast.

Rotation augmentation mimicked various drawing orientations and slight misalignment that could realistically occur during scanning or drafting tasks. By incorporating random rotations within a small range, the relative spatial locations of racks, driveways, and starting points were maintained realistically and thereby enhanced the rotational invariance of the models (Buslaev et al., 2020).

The adjustments for brightness and contrast were done to compensate for variations in document quality, scanner settings, and background lighting. Most CAD drawings had differences in lighting and shading; therefore, the enhancement of these factors assisted in developing models that are robust to well-illuminated and poorly scanned documents.

Around 100 augmented copies were made for each original image. This significantly enlarged the training and validation sets and introduced a huge amount of variability without altering the inherent spatial meanings of the annotated points. Through this heavy augmentation, the models were exposed to a wide range of synthetic noise patterns, geometric translations, and visual degradations during training time, which directly translated to robustness in real-world conditions (Buslaev et al., 2020).

3.5.2 Dataset Splitting

After the augmentation process, it was important to split the dataset correctly to facilitate proper training, validation, and testing processes. For maintaining a fair evaluation, the augmented dataset was properly split into three distinct sets: training, validation, and testing.

All the dataset splitting and label handling were done using custom Python scripts. Stratification guaranteed a balanced and fair evaluation pipeline, ultimately enabling the selection of well-generalizing models outside of the augmented training data. Cross-validation was not employed in this thesis due to the limited size of the original, manually annotated dataset and the extensive use of data augmentation. As highlighted by (Lee et al., 2023), applying standard k-fold cross-validation after augmentation can lead to data leakage, where augmented versions of the same base image appear in both training and validation folds, artificially inflating performance metrics.

3.5.3 YOLO label format Conversion

The annotation format that YOLO uses is specifically designed for single-stage object detectors that regress bounding boxes and class labels directly from input images in a single forward pass (Jocher et al., 2023). In this format, every image has a corresponding `.txt` file in which each object instance is described in one line with five normalized values: object class index (c) and bounding box center coordinates (x, y), and the box width (w) and height (h). Normalization is done relative to image width and height using the formulas:

$$x = \frac{x_{\text{center}}}{W}, \quad y = \frac{y_{\text{center}}}{H}, \quad w = \frac{w_{\text{bbox}}}{W}, \quad h = \frac{h_{\text{bbox}}}{H} \tag{1}$$

where W and H are the width and height of the image in pixels.

During conversion, bounding box coordinates were normalized and clipped to ensure they remained within the $[0,1]$ interval, adhering to YOLO's input expectations (Jocher et al., 2023). Each annotation `.txt` file was matched exactly to the corresponding image filename (for instance, `image001.png` $\rightarrow$ `image001.txt`) to maintain dataset integrity.

Under the conversion process, a tiny 10-pixel synthetic bounding box was drawn around each origin, rack, and driveway point to facilitate its encoding in the YOLO setup. The object classes were numerically encoded (for example, Origin = 0, Rack = 1, Driveway = 2) such that they could be standardized across the dataset. This allows all spatial objects in the CAD layouts, whether point-based or region-based, to be treated uniformly by the YOLO object detection model.

Lastly, the YOLO-structured dataset enabled the model to acquire a complete yet detailed set of warehouse layouts, which is essential for the later stages of SQL schema generation. Furthermore, it took advantage of YOLO's capabilities in accurately detecting several objects simultaneously, thereby attaining real-time processing speeds (Jocher et al., 2023).

3.6 Model Training

The initial training was done using three different models: keypoint detection, Detectron2, and YOLO. However, the keypoint and Detectron2 models performed poorly, with less than 50% accuracy, so we decided to drop them. Then the remaining training process was structured around different YOLO detection models, which gives promising initial results.

3.6.1 YOLO Object Detection Model

YOLO object detection model was used for object classification and localization, i.e., racks and driveways, in warehouse CAD images. We used different yolo models of different versions and sizes to see how the model learn the images. The model parameters are listed as per Table 1.

Table 1

Comparison of YOLO Model Variants

YOLO Variant	Input Size	Parameters (M)	FLOPs (B)
YOLOv8m	640	25.9	78.9
YOLOv8l	640	43.7	165.2
YOLOv8x	640	68.2	257.8
YOLOv9c	640	25.3	102.1
YOLOv10l	640	29.5	120.3
YOLOv11l	640	25.3	86.9

In this project, we used the YOLO base model for object detection. However, to improve performance in crowded layouts and handle class imbalance, we modified the default loss function by implementing a custom loss based on the original `v8DetectionLoss` from Ultralytics. The new loss function integrates Varifocal Loss (VFL), which is more suitable for multi-object scenes. This loss function is used with weighted classes and without weighted classes to test different combinations to find the best model. Along with that, a model(v8x_orig_2048_cw) without any custom loss function is also trained to compare the results.

The training was done at an increased input resolution of 1536×1536 and 2048×2048 pixels to enable more spatial detail for the model to learn small bounding boxes effectively. Each model is trained for 100 epochs using a batch size of 1, which is necessary to accommodate the high memory requirements associated with large input image resolutions (1536×1536 or 2048×2048). All training is conducted on an SLURM-managed GPU cluster node equipped with an NVIDIA A100 80GB PCIe GPU, configured in a 10 GB MIG (Multi-Instance GPU) profile. CUDA acceleration is enabled for all runs, and Automatic Mixed Precision (AMP) is utilized to reduce GPU memory consumption and improve throughput. Experiment management, hyperparameter tracking, and visualization of loss convergence and metric performance over epochs were achieved with Comet.ml. The hyperparameters for different model trainings are listed in Table 2. The Model ID is named after YOLO versions, image resolution, and model size.

Table 2

YOLO Model Hyperparameters Summary

Model ID	Model	Batch	Epochs	Img Size	Box	Cls	HSV H	HSV S	HSV V	Mosaic	Scale	VFL
v8_1536_x	yolov8x.pt	1	100	1536	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true
v8_2048_l	yolov8l.pt	1	100	2048	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true
v8_2048_m	yolov8m.pt	1	100	2048	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true
v8_2048_x	yolov8x.pt	1	100	2048	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true
v8_orig_2048_cw_x	yolov8x.pt	1	100	2048	10.0	1.0	0.0	0.0	0.0	0.5	0.0	false
v9_2048_c	yolov9c.pt	1	100	2048	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true
v9_2048_nw_c	yolov9c.pt	1	100	2048	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true
v10_2048_l	yolov10l.pt	1	100	2048	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true
v11_2048_l	yolo11l.pt	1	100	2048	10.0	1.0	0.015	0.7	0.4	1.0	0.5	true

The hyperparameters listed in Table 2 were applied consistently across all YOLO variants, with configurations such as hue jittering, mosaic augmentation, and scale normalization defined as part of the training setup (see 2.6.1).

In terms of loss configuration:

- The model `v8_orig_2048_cw_x` represents the original YOLOv8x without any modifications to the default loss function, and was trained without VFL.
- The model `v9_2048_c` incorporates both VFL and class-weighted loss, making it suitable for imbalanced class scenarios.
- The model `v9_2048_nw_c` also uses VFL but without class weighting, allowing comparison of the specific effect of weighted loss.

These configurations were selected to observe how VFL, class weighting, and image resolution influence detection performance in dense and semantically rich CAD layouts.

3.7 Custom Loss Function without Class Weights

We modified the default YOLO loss by integrating Varifocal Loss (VFL) as proposed by Zhang et al. (H. Zhang et al., 2021) for classification, replacing the standard Binary Cross-Entropy (BCE) loss. This improves performance in scenarios with class imbalance and overlapping objects. The total loss function in our model consists of three components: bounding box loss, classification loss, and Distance Focal Loss (DFL), following the multi-part design commonly used in YOLO based detectors (Ultralytics, 2023):

$$\mathscr{L}_{\text{total}} = \lambda_{\text{box}} \cdot \mathscr{L}_{\text{box}} + \lambda_{\text{cls}} \cdot \mathscr{L}_{\text{cls}} + \lambda_{\text{dfl}} \cdot \mathscr{L}_{\text{dfl}} \tag{2}$$

where λ_{box}, λ_{cls}, and λ_{dfl} are scalar weights (defined in the model's hyperparameters) that control the contribution of each term.

When Varifocal Loss is enabled, the classification component is computed as:

$$\mathscr{L}_{\text{VFL}} = -\sum_i \alpha_i (1 - p_i)^{\gamma} \log(p_i) \tag{3}$$

This formulation, adapted from Zhang et al. (H. Zhang et al., 2021), addresses the imbalance between positive and negative samples. In this equation, p_i represents the predicted confidence score for sample i, α_i is a weight based on the Intersection-over-Union (IoU) between predicted and ground-truth boxes, and γ is a modulating factor that emphasizes hard examples.

During training, target bounding boxes are scaled to the model's output resolution, and anchor points are generated dynamically. The YOLO dynamic assigner is used for label assignment. Notably, prediction tensors are detached during assignment to prevent unwanted gradient propagation, as recommended by Ultralytics (Ultralytics, 2023). This custom loss function is used in all the models except v8_orig_2048_cw_x and v9_2048_nw_c

3.8 Class-Weighted Custom Loss Function

To further address class imbalance, we extend the Varifocal Loss by incorporating manual per-class weights, a method adapted from the original formulation by Zhang et al. (H. Zhang et al., 2021) and influenced by class-imbalance mitigation strategies such as the Focal Loss proposed by Lin et al. (T. Y. Lin et al., 2017). This approach is further supported by broader findings on the effectiveness of class-weighting in imbalanced deep learning settings (S. Wang et al., 2016).

For instance, the “Origin” class appears much less frequently than “Rack” or “Driveway” in our dataset. This was based on the number of instances of each class in the training data. To compensate, we assign static weights w_c to each class c as follows:

$$w_c = \begin{cases} 2.5 & \text{if } c = \text{Origin} \\ 0.9 & \text{if } c = \text{Rack} \\ 1.0 & \text{if } c = \text{Driveway} \end{cases} \tag{4}$$

These weights are applied directly to the target scores during the computation of the Varifocal Loss:

$$\mathscr{L}_{\text{VFL-Weighted}} = -\sum_i w_{c_i} \cdot \alpha_i (1 - p_i)^{\gamma} \log(p_i) \tag{5}$$

Here, w_{c_i} denotes the class weight associated with the ground-truth label c_i of sample i, p_i is the predicted objectness score, α_i is an IoU-based weighting factor, and γ is a focusing parameter. This formulation increases the contribution of underrepresented classes to the loss, thereby improving model performance in imbalanced multi-class settings.

3.9 Anchor Assignment and Box Decoding

We follow the default YOLO pipeline for anchor assignment and box decoding (Ultralytics, 2023). Anchor points are generated for each output scale using feature map dimensions and strides. During training, these anchors are matched to ground-truth boxes using a dynamic label assigner that considers both spatial proximity and classification confidence. Predicted bounding boxes are then decoded using learned distributions and are evaluated against the ground truth using IoU-based loss terms for both $\mathscr{L}_{\text{box}}$ and $\mathscr{L}_{\text{dfl}}$.

3.10 Training Behavior

The modified loss helps the model learn better in scenes where many objects overlap or when some classes are rare. Since VFL focuses on confident and correct predictions, the model becomes more stable and less biased toward frequent classes (H. Zhang et al., 2021).

Furthermore, a box loss weight of 10.0, much larger than the default, was employed to prefer spatial accuracy to classification when detecting small, close structures. A cosine learning rate schedule was employed to enable smoother convergence and stable generalization in later epochs (Yu et al., 2021). The training loop employed a customized trainer that substitutes the standard operation to include the enhanced loss calculation and enhances inference visualizations. The model was trained on NVIDIA GPUs with CUDA acceleration, and all predictions were logged in structured JSON format to facilitate subsequent SQL evaluation.

3.11 Inference and Postprocessing

Following the training phase, an inference and postprocessing pipeline was developed to transform the predictions of the trained model into structured outputs suitable for spatial computation and the eventual SQL command generation. What follows is an explanation of the object-level and point-level inference operations for warehouse CAD layouts.

3.11.1 YOLO Prediction

The trained YOLO model was used to detect and classify key spatial elements in the warehouse layout dataset, specifically *Racks* and *Driveways*. Inference was performed on the test dataset using an input resolution of 1536×1536 pixels and a confidence threshold of 0.3. This lower threshold was chosen to increase the model's sensitivity, helping it capture subtle or low-contrast features, which are common in CAD-style renderings of warehouse environments.

For each test image, the model produced predictions in the form of bounding boxes, class labels, and confidence scores. The bounding boxes were returned in the format (x, y, w, h), where (x, y) represents the coordinates of the top-left corner, and w and h are the width and height of the detected region, respectively.

Each prediction result was formatted as a dictionary that included the image file name, the predicted class label, the coordinates of the bounding box, and a confidence score. The bounding boxes were represented using the top-left and bottom-right corners in pixel coordinates: (x_1, y_1) and (x_2, y_2). A typical prediction entry looked like this:

```
{
    "image": "image.jpg",
    "class": "Rack",
    "x1": 577.6, "y1": 694.7, "x2": 592.9, "y2": 710.1,
    "confidence": 0.89
}
```

All prediction entries were stored in a list and saved as a single JSON file for SQL generation and spatial layout analysis. This structured format made it easy to process predictions programmatically and ensured that model inference remained decoupled from the post-processing pipeline. Additionally, the predictions were visualized by drawing labeled bounding boxes on the original test images and saved for qualitative inspection.

The inference process was accelerated using a GPU (CUDA) backend to enable high-speed prediction, even on large-scale layouts. All prediction outputs were logged to `Comet.ml` via API integration, as described in the experimental setup(2.8). The resulting JSON file served as the centralized source of detection data, supporting detailed analysis and visualization of the model's performance.

To further improve prediction precision, especially in complex or cluttered layouts, colored dot markers were used in the test images to indicate the location of known objects such as *Racks*, *Origin*, and *Driveways*. These visual cues helped the model distinguish between overlapping or low-contrast elements that are otherwise difficult to separate in CAD renderings.

3.11.2 Pixel Distance Calculation

After completing the object detection step, we computed the center point of each detected object using the bounding box coordinates provided by the model. Each bounding box is defined by its top-left (x_1, y_1) and bottom-right (x_2, y_2) corners. The center (x_c, y_c) of the bounding box was calculated as:

$$x_c = \frac{x_1 + x_2}{2}, \quad y_c = \frac{y_1 + y_2}{2} \tag{6}$$

These center coordinates were used to represent the spatial location of detected objects such as *Racks* and *Driveways*.

To analyze the layout and validate the predictions against known CAD dimensions, we measured the pixel distance along the x-axis between the center of each detected object and a fixed reference point, called the origin $O = (x_o, y_o)$. The horizontal pixel distance D_p between the origin and an object $P = (x_p, y_p)$ was computed as:

$$D_p = |x_o - x_p| \tag{7}$$

This approach assumes that the CAD layout is primarily aligned along the horizontal (x) axis, which is typical in warehouse environments where racks are arranged in rows.

To convert these pixel distances into real-world units (millimeters), a scaling factor s in mm/px was applied. The predicted distance in millimeters D_{mm} was then calculated as:

$$D_{mm} = D_p \cdot s \tag{8}$$

Finally, to validate the detection accuracy, we compared the predicted distances with ground-truth distances from the CAD layout. The absolute deviation between the predicted and CAD distances was calculated to quantify the model's precision. These comparisons were recorded for further analysis and tabulated for evaluation.

3.11.3 Manual Validation

A custom-built pixel click tool was developed using OpenCV to quantify the precision of the measurements that the model would predict. The interactive tool enabled human users to manually click two points on the CAD image usually the origin and the closest rack and compare the computed pixel distance with the estimation made by the model. The tool applies the same millimeter-per-pixel scale S for translation and then computes the deviation δ between the model's estimation and the manual measurement:

$$\delta = |D_{\text{CAD}} - (D_p \times S)| \tag{9}$$

The use of this manual validation method enabled a semi-automated review of geometric integrity, serving as a reliability check to ensure the efficacy of the detection and scaling pipeline. Deviations were meticulously recorded and compared to assess the accuracy of the system. There is a 50mm tolerance level applied when converting pixel distance to real distance as per CAD scale. 3 full warehouse diagrams are chosen to perform manual ground truth analysis. The predicted points and actual points of interest are calculated and evaluated based on the count. This will help to understand how many points are missed and extra.

The combination of model-based computation and human-in-the-loop verification achieved numerical precision and qualitative certainty. The iterative review played a critical role in validating that SQL schemas derived from CAD designs were geometrically compatible and conformable to industrial warehouse requirements.

3.12 Model Validation and Fine-Tuning

To improve detection performance, the model was fine-tuned by adjusting augmentation settings and rebalancing the training data. Rotations were limited to within $\pm 10^{\circ}$, and underrepresented classes like driveways were oversampled to reduce class imbalance. Additionally, key YOLO hyperparameters such as box loss weight and confidence threshold were tuned to enhance localization and reduce false negatives.

3.13 Evaluation Metrics and Scoring Strategy

To objectively assess model performance, we used three standard metrics in object detection: mean Average Precision(mAP), Recall, and Precision. These metrics evaluate both the classification and localization performance of the detector and are widely adopted in benchmarks such as Pascal VOC and COCO (Everingham et al., 2010; T.-Y. Lin et al., 2014). This provides classification performance metrics for the various components identified in the CAD drawings.

- **Mean Average Precision (mAP@50:95)**: This metric evaluates how accurately the model detects objects across multiple Intersection over Union (IoU) thresholds, typically from 0.50 to 0.95 in steps of 0.05.
- **Recall**: Recall measures the ability of the model to detect all relevant instances. It is defined as the ratio of true positive detections to the total number of ground truth instances.
- **Precision**: Precision quantifies the correctness of the model's predictions, defined as the ratio of true positives to all predicted positives.

3.14 Model Selection Strategy

To identify the most suitable model for the proposed system, a structured selection process was followed. This involved evaluating the selected models across three main criteria:

- Quantitative Metrics: Standard evaluation metrics such as precision, recall, mAP@50, and mAP@50:95 were recorded on both validation and test datasets to assess baseline detection performance(see 3.13). This is done against the test and validation dataset.
- Manual Evaluation: Each model was further tested on full warehouse diagrams that were not part of the training, validation, or test datasets. The predicted points were compared against manually annotated ground truth, measuring the number of matched, missed, and extra detections(see 3.11.3).
- Resource Efficiency: In addition to accuracy, the model's training time, memory usage, and CPU/GPU load were tracked to understand how demanding it is to run. This is important because, in the future, the model needs to handle larger and more complex CAD drawings. While this thesis used images up to 2048×2048 pixels, real-world layouts are much larger and sometimes over 11,000 pixels wide, which would make training more challenging. Keeping the model efficient ensures it can be scaled and used in practical scenarios without requiring excessive resources.

By combining these three layers of evaluation, the aim was to find a model that not only performs well on metrics but also generalizes effectively to unseen layouts and remains computationally feasible for practical deployment.

3.15 SQL Generation

From the model prediction JSON, the final output SQL is generated using a Python script. This script will find the origin first by checking the class names in the model prediction output. Then it will find and sort all the racks in ascending order of shortest distance to the origin on the x-axis. This is because the CAD diagrams are horizontally aligned and the racks are named with consecutive numbers from left to right. The positions of the racks are converted to actual distances by the CAD scale as mentioned in 3.11.2. Then, an insert script is created based on the positions of the sorted racks.

3.16 Ethical Consideration

This thesis aims to automate spatial information extraction from warehouse CAD diagrams using deep learning-based computer vision techniques. Although the study does not involve personal or sensitive human data, ethical dimensions such as data ownership, transparency, and the effects of automation on the workforce are brought to the agenda.

First of all, all CAD data used are synthetically generated or anonymized and licensed datasets by CL. No data containing customer-specific or confidential commercial information was used. This complies with the European Union General Data Protection Regulation (GDPR, Regulation EU 2016/679). The diagrams used here don't reveal actual warehouse details and the location, which makes them anonymous.

Secondly, automatically generating SQL commands from spatial plans brings with it important ethical discussions in terms of human control and error liability. To reduce these risks, the generated SQL outputs were verified by experts, and manual control mechanisms were added to the process (Binns et al., 2018).

In addition, although artificial intelligence applications in warehouse automation increase efficiency, they may lead to the reduction of some manual labor roles. Therefore, the developed systems should support human skills and be designed by the principles of transparency and fairness (Floridi et al., 2018; IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, 2019).

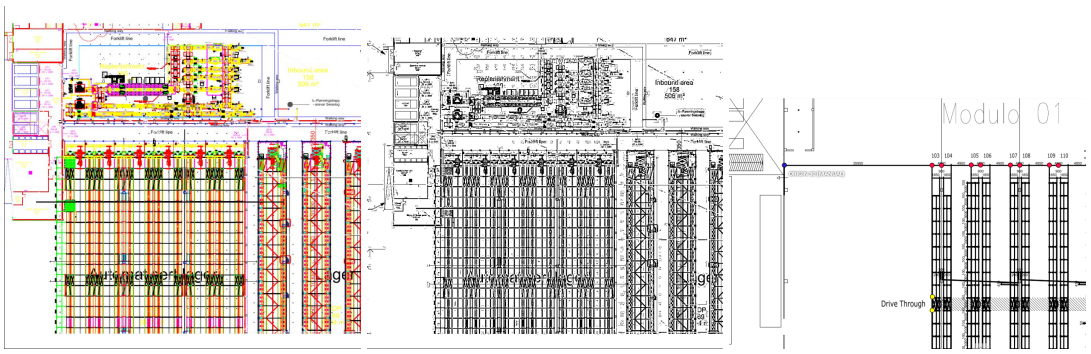
Finally, the methods developed in the thesis were carried out in line with the European Commission's principles of trustworthy artificial intelligence: legality, ethical purpose, and robustness (Smuha, 2019).

4 Results and Evaluation

Here we discuss the results from various methodologies(see Figure 2) used in this thesis, starting from data annotation.

4.1 Annotation and Color Transformations

After the color transformation, the colored CAD images are converted to a black and white image as in Figure 4b. Figure 4c illustrates the point-based annotations applied to the black and white images using the CVAT tool. Classes such as origin, racks, and driveways are represented with colored dots (blue, red, and yellow, respectively). Other than class annotations, the training image will be black and white.



(a) Original colored CAD drawing *(b)* Black & white converted *(c)* Annotated CAD drawing

Figure 4. CAD drawings transformation for training

These labeling strategies, especially the addition of color cues and point-based labeling, directly contribute to answering Subquestion 2 (RQ1.2, see section 1.2), which focuses on accurately extracting spatial relationships and pixel coordinates from annotated CAD images. Accurate distance calculations are made possible by providing visual references for starting points, racks, and driveways.

4.2 Data Augmentation Samples

To enhance generalization, augmented variants were generated using the Albumentations library. Figure 5 presents augmented samples exhibiting rotation, flipping, brightness, and contrast changes.

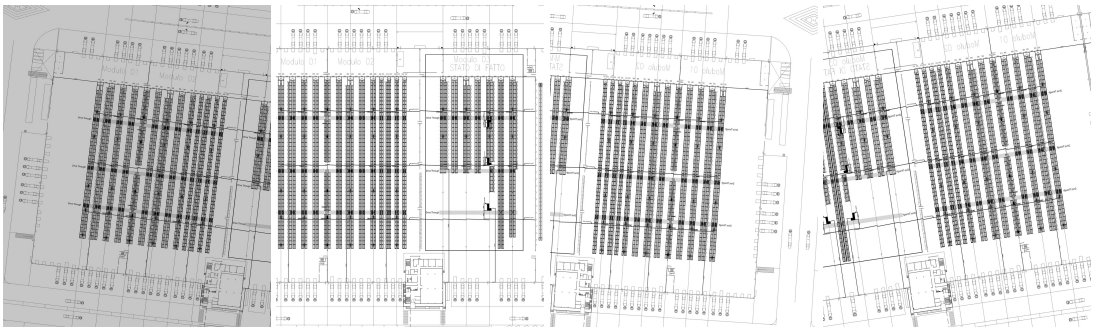


Figure 5. Data Augmentation Samples

4.3 YOLO Model Training Environment and Resource Summary

A total of nine different YOLO models were trained as part of this study, as summarized in Table 3. Model training durations varied between 5.3 and 15 hours, depending on the model complexity, as indicated by the total number of parameters and the resolution of the input as per Table 2 and Table 1. The lightest model (`v8_2048_m`) completed in approximately 5.3 hours, while the most computationally demanding configuration (`v8_orig_2048_cw_x`) required nearly 15 hours. The average memory usage during training ranged from 4.6 GB to 8.9 GB across models, with corresponding CPU loads varying between 6% and 96%. These values reflect the combination of data loading overhead, model size, and system-level multi-threading behavior. All training runs utilized eight data loader workers and were configured with automatic optimizer selection, which defaulted to AdamW in most cases.

Table 3

Training Time, Memory Usage, and CPU Load for YOLO Models

Model	Training Time (hrs)	Memory Usage (GB)	CPU Load (%)
v8_2048_m	5.30	4.62	35.64
v8_2048_l	7.67	6.52	16.87
v11_2048_l	7.96	7.96	96.68
v9_2048_c	8.20	6.77	84.43
v9_2048_nw_c	8.21	6.75	66.59
v8_1536_x	8.56	5.30	29.66
v10_2048_l	9.54	8.87	41.04
v8_2048_x	11.70	7.94	62.40
v8_orig_2048_cw_x	14.99	7.92	6.26

The training infrastructure described in this section supports the experimental process designed to answer the Main RQ (RQ1, see section 1.2), which concerns the identification of elements in CAD diagrams using computer vision techniques. In this context, different YOLO versions were tested and compared to determine the most effective object detection model.

4.4 Training Results

The evaluation metrics of validation and test predictions are listed according to Table 4. Since the task involved detecting specific points rather than full objects, there weren't many reliable model options available. YOLO showed the best results early on, so we focused on it and experimented with different YOLO variants to fine-tune performance. Here, the YOLO Version 11 large model trained with 2048 image dimensions has the best performance based on the different metrics mentioned in 3.13.

Table 4

Validation and Test Metrics per Model

Model	Precision_val	Recall_val	mAP50_val	mAP50_95_val	Precision_test	Recall_test	mAP50_test	mAP50_95_test
v11_2048_l	0.9979	1.0000	0.9950	0.8545	0.9997	1.0000	0.995	0.8817
v8_2048_l	0.9945	0.9999	0.9950	0.8288	0.9996	1.0000	0.995	0.8877
v8_orig_2048_cw_x	0.9987	0.9999	0.9950	0.8256	0.9985	1.0000	0.995	0.8532
v9_2048_c	0.9982	0.9997	0.9950	0.8182	0.9997	1.0000	0.995	0.8508
v9_2048_nw_c	0.9982	0.9997	0.9950	0.8182	0.9997	1.0000	0.995	0.8508
v8_2048_x	0.9972	0.9852	0.9901	0.8146	0.9987	1.0000	0.995	0.8621
v8_2048_m	0.9939	0.9617	0.9950	0.8095	0.9989	1.0000	0.995	0.8808
v10_2048_l	0.9961	0.9999	0.9950	0.8163	0.9921	0.9964	0.995	0.8433
v8_1536_x	0.9936	0.9999	0.9950	0.7738	0.9988	1.0000	0.995	0.8719

The validation and test metrics showed that most YOLO models performed similarly well, with only minor differences between them. To further evaluate the results, we manually tested predictions on three complete warehouse diagrams and compared them with the ground truth.

Table 5

Manual Evaluation of YOLO Models on 3 Full Warehouse Diagrams

Model	Ground Truth Total	Predicted Total	Matched	Missed	Extra
v9_2048_nw_c	83	83	82	1	1
v9_2048_c	83	83	82	1	1
v8_orig_2048_cw_x	83	82	81	2	1
v10_2048_l	83	78	78	5	0
v11_2048_l	83	77	77	6	0
v8_2048_l	83	78	76	7	2
v8_2048_x	83	75	74	9	1
v8_2048_m	83	70	69	14	1
v8_1536_x	83	65	65	18	0

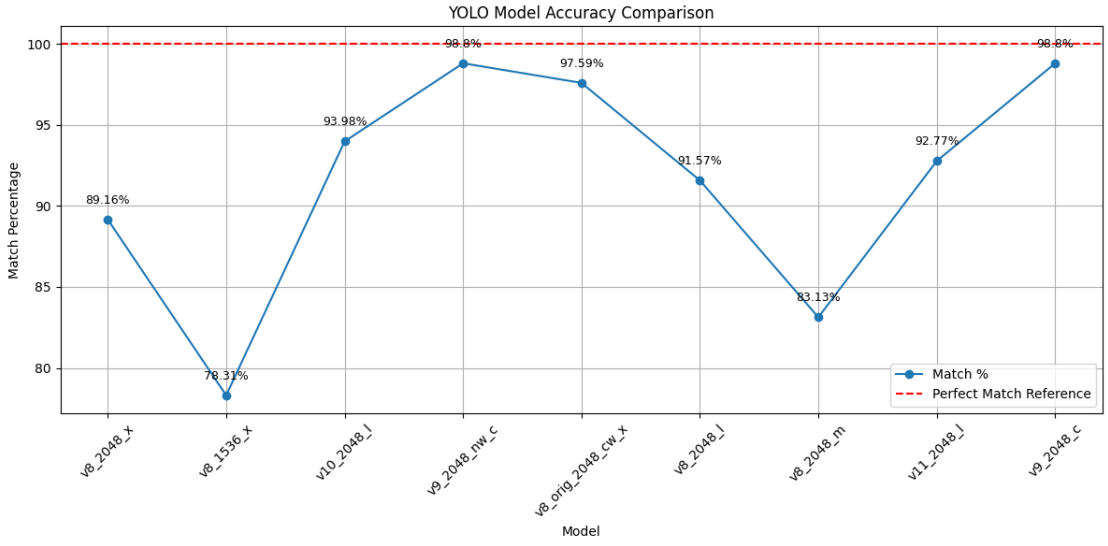


Figure 6. Prediction match percentage to ground truth

We evaluated each YOLO model on three complete warehouse diagrams by comparing the model predictions with the ground truth point annotations. The YOLOv9c_2048 and YOLOv9c_2048_weighted models performed the best, each correctly matching 82 out of 83 points, achieving a 98.8% match rate. The next best was YOLOv8x_original_1536, with a 97.59% match rate. The Figure 6 represents the match percentage of each model.

Models like YOLOv10l_2048 and YOLOv11l_2048 also performed reasonably well, with match rates above 90%. However, models, such as YOLOv8x_1536 and YOLOv8m_2048 struggled more, missing a larger number of ground truth points and scoring below 85%. YOLOv8m_2048 is the only medium-sized model we used, and YOLOv8x_1536 has a lower resolution dataset compared to other models, which can be a reason for the low performance.

In general, the results show that while most models perform fairly well, YOLOv9c_2048 consistently delivers the highest accuracy and is most reliable for this task. From Figure 7, the predicted bounding boxes are shown.

These model performance comparisons also contribute to answering Sub-Research Question 1 (RQ1.1, see Section 1.2) by revealing the most effective computer vision models for detecting warehouse elements in CAD-based layouts.

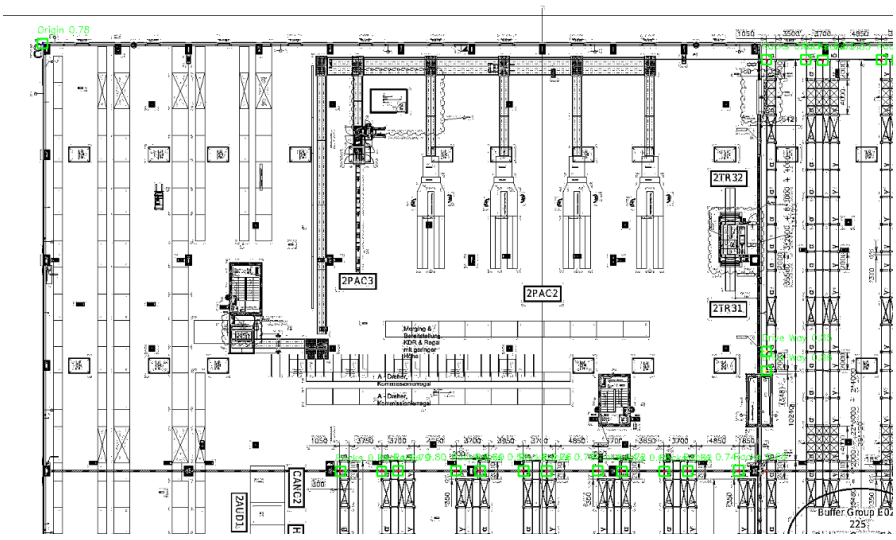


Figure 7. Prediction of racks

4.5 Distance Validation

Figure 8 shows the predicted locations of the origin and the first rack (Rack 103) in a test image. According to the engineering drawing, the actual distance between these two points is 35,950 mm.

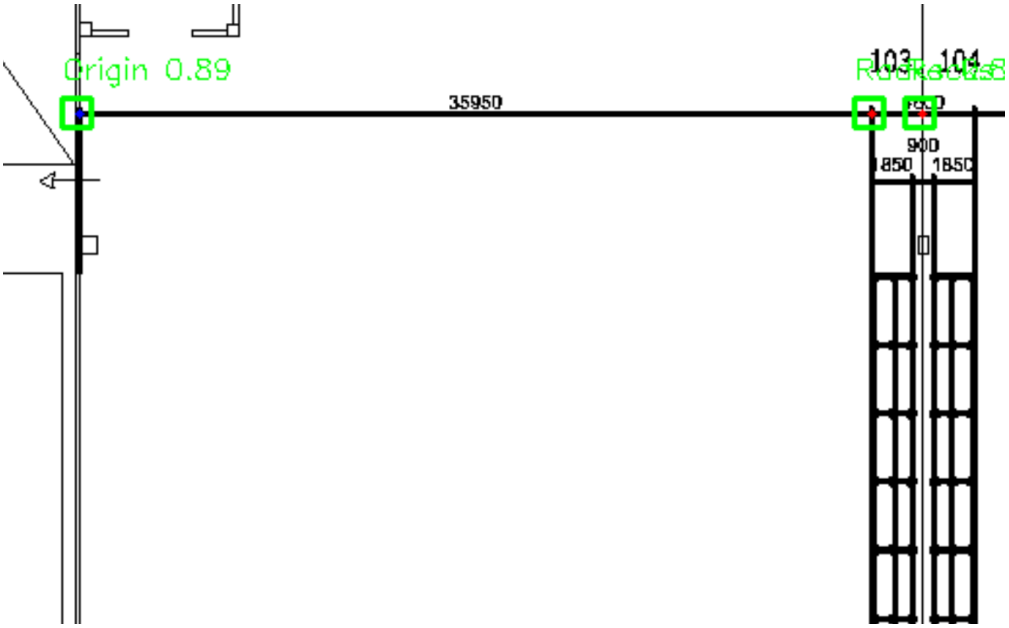


Figure 8. Prediction or Origin and Racks

From the prediction output, the coordinates of the origin are (398.97, 669.15), and the coordinates of Rack 103 are (807.45, 669.43). The horizontal (X-axis) pixel distance between them is approximately 408.48 pixels.

Using a manually calculated scale factor of 87.90 pixels per meter(3.11.3), and applying the distance conversion equation 9, the estimated distance is 35,905.40 mm. This results in an error of only 44.61 mm

from the ground truth, showing that the prediction is highly accurate.

To generate clean and consistent coordinates for rack placement, a rounding tolerance of 50 mm was applied during SQL generation. This means that each computed position is rounded to the nearest multiple of 50 mm, effectively snapping values to either `.000` or `.050` endings. This approach not only simplifies the output but also aligns with common warehouse layout standards.

The final coordinates were first scaled from pixel space to real-world units using a fixed conversion factor, and then rounded using this 50 mm tolerance. The resulting SQL insert statements are shown in Listing 1. These SQL statements are generated directly from the prediction output by applying equation 8, applying the tolerance, and finding the center of the bounding box by applying equation 6.

Listing 1: Sample SQL Insert Statements for Racks

```
INSERT INTO racks (id, x, y) VALUES (103, 35900.000, 0.000);
INSERT INTO racks (id, x, y) VALUES (104, 38150.000, 0.000);
INSERT INTO racks (id, x, y) VALUES (105, 45350.000, -50.000);
INSERT INTO racks (id, x, y) VALUES (106, 47650.000, -50.000);
```

4.6 Impact of Annotation and Preprocessing

The quality of the training process and resulting model performance were influenced by the annotation and preprocessing steps. The initial transformation of CAD diagrams from colored to black-and-white format simplified the visual input and helped reduce noise from non-relevant design elements. The models were able to concentrate more on spatial elements like racks, origins, and roadways because of this grayscale conversion, which highlighted the geometric structure of warehouse layouts.

Point-based annotations contributed to precise training by marking the exact pixel locations of relevant elements. These annotations were class-specific and color-coded, which helped the object detector distinguish between similar-looking features. These points were so relevant as the rest of the image is back and white, and this is colored. By minimizing ambiguity in the training labels, the model learned to detect classes more reliably.

Data augmentation also played a key role in improving model robustness. Techniques such as flipping, rotation, and brightness adjustments—applied, introduced variation into the dataset (Buslaev et al., 2020). This helped reduce overfitting and ensured that the models could generalize better to unseen CAD diagrams. This worked to an extent, as we can see the newly tested CAD images were predicted with more than 90% accuracy(see Table 5). Also, augmented samples preserved the structural integrity of the layout, which was important for maintaining consistent spatial relationships during training. Since there weren't enough independent CAD drawings for training, this method helps to create enough data for model training.

In addition, the handling of image dimensions contributed to the stability of training. All training images were resized to high resolutions (1536 to 2048 pixels) to preserve detail, particularly for the colored points. But, it was also observed that when the aspect ratio of original test images differed significantly from training images, performance could decline. This can likely be due to spatial distortion introduced during resizing as mentioned in (Tong & Wu, 2024).

Overall, annotation precision and consistent preprocessing were critical to achieving stable and high-performing training results. It is also noted that the images without color coding performed poorly, and sometimes none of the keypoints are detected. Which means that the model learned color-based keypoints.

4.7 Justification for Best Model Selection

Based on the selection strategy described in Section 3.14, the YOLOv9_2048_c model was identified as the most suitable candidate for integration into the proposed CAD diagram interpretation pipeline. While multiple YOLO variants—including YOLOv11_2048_l and YOLOv8_2048_l—exhibited strong validation metrics, the final selection is not based on quantitative results but also real-world consistency, robustness, and computational efficiency.

Quantitatively, YOLOv9_2048_c demonstrated competitive performance across evaluation metrics. As shown in Table 4, it achieved near-perfect test precision (0.9997), recall (1.0000), and mAP@50 (0.995), along with a strong mAP@50:95 of 0.8508. Although a few models marginally outperformed it on certain metrics, such as mAP@50:95, those models fell short in qualitative assessments or demanded significantly higher computational resources.

In full-layout manual evaluations (Table 5), YOLOv9_2048_c outperformed all other models by correctly detecting 82 out of 83 annotated elements, resulting in a match rate of 98.8%. The same YOLOv9c weighted model(YOLOv9_2048_nw_c) also has the same metrics, which gives a conclusion that class imbalance didn't impact the model. The class weighted and nonweighted models perform almost similar way. This practical evaluation was critical in confirming that the model could generalize well to real warehouse diagrams, not just cropped training samples. Moreover, both the weighted and non-weighted variants of this model produced identical results in manual tests, indicating strong stability to changes in class distribution.

Training performance was also stable and efficient for these models compared to larger models such as YOLOv11_2048_l, which required up to 15 hours and used more than 8 GB of memory, YOLOv9_2048_c maintained high detection accuracy with moderate training time and memory usage (see Table 3). This trade-off makes it more suitable for scalable deployment, especially in settings with limited computational capacity.

In summary, YOLOv9_2048_c and YOLOv9_2048_nw_c provided a well-rounded solution across multiple dimensions—accuracy, robustness, training efficiency, and generalization. These qualities make it the most appropriate choice for deployment in the proposed system for automatic extraction of spatial data from warehouse CAD drawings. The results from the evaluation metrics(Table 4,5) strongly support this claim. The final CAD-to-SQL pipeline is based on the YOLOv9c model described below.

4.8 Final Pipeline Overview

The final pipeline developed in this thesis transforms warehouse CAD drawings into structured SQL commands through a series of semi-automated steps. It begins with preprocessing the CAD images, which involves converting them to black-and-white format and adding color-coded points to represent the position of key components such as racks, driveways, and the origin. These images are then processed using a pre-trained YOLOv9c model that detects components such as racks, driveways, and the origin in the new CAD layouts. After detection, the coordinates of the identified elements are scaled using the scale of the CAD drawing and the pixel coordinates. These positions are then formatted into SQL `INSERT` statements that can be used to populate warehouse databases.

A visual summary of this pipeline is shown in Figure 9. The trained YOLOv9c model achieved a precision of 0.9997, a recall of 1.0000, and a mean Average Precision (mAP@50) of 0.995 during evaluation. In full-layout testing, it correctly matched 98.8% of annotated elements, demonstrating strong performance and generalization to unseen CAD diagrams.

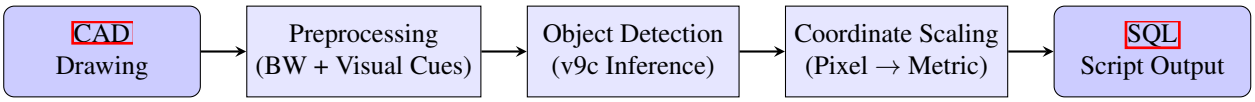


Figure 9. Final CAD-to-SQL inference pipeline used for semi-automated layout processing

5 Discussion

This chapter interprets the key findings about the research questions and objectives. It also discusses the limitations of the study, reflects on the development process, and the ethical aspects of deploying AI in warehouse automation.

5.1 Qualitative Error Cases and Model Limitations

While the YOLO models performed well overall, a few limitations were observed during manual evaluation. For instance, in some cases, the model occasionally failed to detect certain racks or misclassified elements. In some test images, the predicted origin was slightly offset, which affected the relative coordinates of all subsequent racks. These errors typically occurred in diagrams with heavy annotation noise and unconventional layout structures.

One other drawback of the whole model prediction is the dimensions of the training images. All the training images are of near-square images, while real-world CAD drawings are more rectangular. Which will be automatically compressed by the model to adjust to the model dimensions. Even though we trained with very high resolution(1536-2048), some of the real CAD drawings are of a range 11000x7000. This makes the model adjust the image size, and thus the output is of low quality. To train images of these dimensions, a GPU with higher performance and memory is needed.

Another observation from the results is that class imbalance didn't affect the model prediction, as the YOLOv9c model with and without the class-weighted loss function performed the same way, producing almost similar results.

Figure 10. Example of a failure case where several rack positions (marked in red) were missed by the model. This input is from a completely new CAD drawing not seen during training or validation.

Figure 10 shows a failure case where the model failed to detect multiple racks, which are marked manually in red. This test image comes from an entirely new CAD layout that was not part of the training or validation datasets. The model successfully predicted the origin and some nearby racks, but missed many others, particularly those further along the horizontal axis. This highlights a key limitation of the current model: its reduced generalization ability when faced with unseen or structurally different warehouse layouts. A contributing factor may be the significant difference in image dimensions between training and test samples. Larger test images with different aspect ratios may introduce spatial distortions during resizing, negatively impacting detection performance. This limitation aligns with findings by Qiu et al. (2019), who emphasize that altering aspect ratios during preprocessing can cause shape distortions that

hinder accurate object localization, particularly in multi-scale object detection tasks.

This highlights the need for more diverse CAD drawings for training, which helps the model learn new information. This particular drawing is more dense and has many dark sections compared to the training image, which can be a reason for the failed detection.

5.2 RQ-Based Interpretation of Results

This section interprets the results with the research questions defined in Section 1.2.

RQ1 (1.2): *How can computer vision techniques be used to identify and extract spatial information from CAD diagrams?*

The experiments conducted in this study show that deep learning-based object detection models can be effectively applied to extract spatial information from warehouse CAD layouts. Through a pipeline involving image preprocessing, annotation, model training, and post-processing. The trained models were able to detect key warehouse components with high precision. The selected model, YOLOv9_2048_c, demonstrated strong generalization to real-world layout diagrams and supported tasks such as coordinate extraction and SQL generation. This confirms the feasibility of using computer vision for spatial data extraction in CAD environments.

RQ1.1 (1.2): *What computer vision techniques can be used for detecting key warehouse components in CAD-based layouts?*

Several computer vision models were evaluated, including keypoint detection, Detectron2, and YOLO. Among these, the YOLO models significantly outperformed the others. YOLOv9_2048_c was identified as the most reliable model based on a combination of high precision (0.9997), recall (1.0000), and firm performance in manual evaluations. It achieved a 98.8% match rate on unseen full-layout diagrams with colored keypoints, which is very promising in a real-world scenario. These results indicate that compact and fine-tuned YOLO architectures are particularly effective for detecting the pixel positions of different warehouse components when proper marking is added.

RQ1.2 (1.2): *How can spatial relationships and pixel coordinates of detected elements be accurately extracted from annotated CAD images?*

The trained model outputs bounding boxes around the colored points. After inference, the detected pixel coordinates were extracted directly from the center points of the model's predictions as per 4.5. To derive spatial relationships, the coordinates of each detected element were compared against a defined origin point within the image. These pixel-level distances were then converted into real-world measurements using scale factors derived from the original CAD layout as in 4.5. This pipeline of preprocessing, training, and postprocessing proves there is a strong relationship with the pixel coordinates of detected elements and the real-world position of these elements in the warehouse.

5.3 Limitations of the Study

Despite the promising results obtained in this thesis, there are some limitations regarding generalizability and practical applications of the proposed method. Firstly, the thesis is based on a limited set of 30 synthetically generated CAD diagrams, and this dataset is expanded with augmentation techniques. Although this method increases the volume of training data, it does not fully reflect the diversity and complexity of real warehouse layouts used in different industries. Therefore, the ability of the model to generalize to previously unseen or unusual structural diagrams remains limited.

Secondly, the labeling process was manual, and only point-based labels were used for three categories (origin, racks, driveways). This limited label variety limits the system's scalability to complex warehouse representations. Elements such as safety zones, wall structures, or machine areas, which are frequently encountered in real-world scenarios, were not addressed in this thesis.

Another limitation is the dependence on color-coded annotation points. Although the black-and-white preprocessing process significantly increased the model accuracy, current diagram standards may not include such color coding. This necessitates additional preprocessing steps to apply the method in the broader area. Since this is a digital drawing done with special care, adding color-coded annotation points during the drawing step will help to reduce the time, and overall time for adding this to a big warehouse diagram is less than 30 minutes.

Future work should focus on increasing the data diversity, adding more annotation classes, and integrating more robust inference mechanisms. Such improvements will increase the system's reliability and generalizability at an industrial scale.

5.4 Reflections and Lessons Learned

This thesis has provided valuable insights into the technical knowledge of AI-aided CAD diagram interpretation and the broader research process and system design. One of the most important lessons is the critical role of iterative trial-and-error and preprocessing processes in deep learning applications. Early model training attempts with colored CAD drawings failed to demonstrate that even small inconsistencies in data representation can seriously impact model performance. This has highlighted the importance of controlled data preparation, appropriate preprocessing, and domain-based adaptations.

Another important conclusion is unlike natural images, the digital drawings need different logic to calculate the positions of various objects. Natural images, with the help of camera properties, the positions of detected objects can be calculated. But for digital drawings, that advantage is not there. Here, we need to depend on pixel distances and scales used in the drawing, which makes it more reliable if the scales are accurate.

The importance of planning and scope control has been clearly understood from a project management perspective. It has been seen that goals such as expanding labeling categories or incorporating real-time system integration in the early stages can cause the project to lose focus. Accepting such technical and strategic limitations has contributed to the development of project leadership skills.

Finally, this thesis provided an opportunity to understand better the role of applied AI in industrial systems. It revealed the importance of interdisciplinary knowledge, such as CV, relational databases, and WMS. These gains will provide a solid foundation for future academic and professional studies.

5.5 Ethical Consideration

This thesis aims to automate spatial information extraction from warehouse CAD diagrams using deep learning-based computer vision techniques. Although the study does not involve personal or sensitive human data, ethical dimensions such as data ownership, transparency, and the effects of automation on the workforce are brought to the agenda.

First, all CAD data used are synthetically generated or anonymized and licensed by Consafe Logistics AB. No data containing customer-specific or confidential commercial information was used. This complies with the European Union General Data Protection Regulation (GDPR, Regulation EU 2016/679). The diagrams used here don't reveal actual warehouse details and the location, which makes them anonymous.

Secondly, automatically generating SQL commands from spatial plans brings with it important ethical discussions in terms of human control and error liability. To reduce these risks, the generated SQL outputs were verified by experts, and manual control mechanisms were added to the process (Binns et al., 2018).

In addition, although artificial intelligence applications in warehouse automation increase efficiency, they may lead to the reduction of some manual labor roles. Therefore, the developed systems should support human skills and be designed by the principles of transparency and fairness (Floridi et al., 2018; IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems, 2019).

Finally, the methods developed in the thesis were carried out in line with the European Commission's principles of trustworthy artificial intelligence: legality, ethical purpose, and robustness (Smuha, 2019).

5.6 Reliability and Validity

Although the model demonstrated high precision and recall on unseen diagrams, the small dataset limits its generalizability. The manual annotations may introduce some label noise. However, the evaluation metrics and ground truth match rate validation support the system's internal reliability. External validity should be further tested on diverse real-world layouts.

6 Conclusion

6.1 Summary of Results

This thesis developed a hybrid system combining deep learning-based object detection with mathematical scaling logic to automate the conversion of CAD based warehouse layouts into structured SQL commands. The pipeline (Figure 9) exhibits high accuracy in extracting spatial and semantic information from CAD diagrams.

The YOLOv9c model, trained at a resolution of 2048^2, achieved strong detection performance with a precision of 0.9997, recall of 1.0000, and a mean Average Precision (mAP@50) of 0.995. While evaluating a complete layout CAD drawing, this model reached a 98.8% match rate, missing only one component.

The preprocessing strategies played a key role in model success by converting CAD drawings into the black and white format, and introducing color-coded reference cues proved critical. Without this step, the model failed to produce meaningful results.

The results confirmed that the proposed system(Figure 9) effectively addressed the core research questions related to object detection, spatial coordinate scaling, and database population. The pipeline reliably converted CAD based visual information into usable SQL format with high accuracy. This demonstrates that the project fulfills the objectives of the thesis.

6.2 Scientific and Practical Contributions

This thesis makes a significant contribution to the field of AI-aided warehouse digitization by addressing a critical challenge in industrial applications. This topic remains underexplored in the current literature: the automatic extraction of spatial and semantic information from engineering diagrams and their integration into predefined database structures. While numerous studies focus on object detection from natural images or the automatic generation of database schemas, direct conversion of engineering diagrams into structured data remains limited and typically constrained to specific domains.

One of the main scientific contributions of this thesis is the development of an idea that can be utilized to determine the exact position of an element in an engineering drawing, which can be converted to a structured format. This workflow combines deep learning-based object detection, spatial post-processing logic, and steps to transform pixel-level visual information into a relational database structure. The presented framework is not limited to warehouse plans only, but can also be adapted to different areas such as factory layout design, Building Information Modeling (BIM) systems, and other technical schematics.

From a practical perspective, the proposed method automates spatial data entry in WMS systems, reducing the reliance on manual processing and minimizing human errors. It also provides a strong foundation for developing real-time digital twin systems by enabling machine-readable information from engineering documents. Thus, this thesis presents a holistic and scalable approach combining applied AI, industrial automation, and CV aided spatial reasoning.

Nevertheless, the system's current scope is primarily validated in a controlled setting with limited data, and further real-world testing with diverse training data is required to realize its cross-domain potential fully.

6.3 Future Work

First, although the YOLOv8–YOLOv11 models successfully detected CAD elements such as racks and driveways, transformer-based detection models (e.g. DETR or DINO) can be integrated for better performance in complex or noisy diagrams (Carion et al., 2020; S. Zhang et al., 2022). One major next step involves the detection of modules that contain multiple racks or embedded storage units. These grouped components needed a different strategy than the current pipeline.

Secondly, the system used in the generation of SQL commands is currently based solely on geometric rules and is straightforward. In the future, the integration of graph-based inference techniques may provide more robust results, especially for complex relationships such as object grouping based on contextual locations or proximity-based passage detection (Marcus, 2020).

Thirdly, the thesis focused only on warehouse plans. However, the developed methodology can also be adapted to other engineering diagrams, such as factory plans, city maps, or Building Information Modeling (BIM) documents, which have a scale. Domain adaptation or few-shot learning approaches can be evaluated for such cross-domain generalization.

Fourth, the labeling in the current study was done completely manually. In the future, semi-supervised or active learning techniques can be used to reduce the cost of data labeling and increase the dataset more efficiently (Zoph et al., 2020).

References

Alicke, K., Rexhausen, D., & Seyfert, A. (2017). Supply chain 4.0 in consumer goods. *Mckinsey & Company*, *1*(11), 1–11.

Altarazi, S. A., & Ammouri, M. M. (2018). Concurrent manual-order-picking warehouse design: A simulation-based design of experiments approach. *International Journal of Production Research*, *56*(23), 7103–7121.

Bao, C., Cao, J., Hao, Q., Cheng, Y., Ning, Y., & Zhao, T. (2023). Dual-yolo architecture from infrared and visible images for object detection. *Sensors*, *23*(6), 2934.

Bhanbhro, H., Hooi, Y. K., Hassan, Z., & Sohu, N. (2022). Modern deep learning approaches for symbol detection in complex engineering drawings. *2022 International Conference on Digital Transformation and Intelligence (ICDI)*, 121–126. https://doi.org/10.1109/ICDI57181.2022.10007281

Bhandari, B., & Manandhar, P. (2023). Integrating computer vision and cad for precise dimension extraction and 3d solid model regeneration. *Machines*, *11*(12), 1083. https://doi.org/10.3390/machines11121083

Binns, R., Veale, M., Van Kleek, M., & Shadbolt, N. (2018). 'it's reducing a human being to a percentage': Perceptions of justice in algorithmic decisions. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*.

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. *Information*, *11*(2), 125.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. *European Conference on Computer Vision (ECCV)*.

Comet ML. (2024). Comet: Ml experiment tracking [Accessed: 2025-04-15].

Creek, T., & Mullins, B. E. (2022). Analysis of image thresholding algorithms for automated machine learning training data generation. *International Conference on Cyber Warfare and Security*, *17*(1), 449–458.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, *1*, 886–893. https://doi.org/10.1109/CVPR.2005.177

Encarnacao, J. L., Lindner, R., & Schlechtendahl, E. G. (2012). *Computer aided design: Fundamentals and system architectures*. Springer Science & Business Media.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Floridi, L., Cowls, J., Beltrametti, M., Chatila, R., Chazerand, P., Dignum, V., Luetge, C., Madelin, R., Pagallo, U., Rossi, F., et al. (2018). Ai4people—an ethical framework for a good ai society: Opportunities, risks, principles, and recommendations. *Minds and Machines*, *28*(4), 689–707.

Flynn, P., & Jain, A. (1989). Cad-based computer vision: From cad models to relational graphs. *Conference Proceedings., IEEE International Conference on Systems, Man and Cybernetics*, 162–167 vol.1. https://doi.org/10.1109/ICSMC.1989.71272

Girshick, R. (2015). Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1440–1448. https://doi.org/10.1109/ICCV.2015.169

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 580–587. https://doi.org/10.1109/CVPR.2014.81

Glyph & Cog, L. (2025). *Pdftoppm manual page* [Version 4.05]. https://fossies.org/linux/xpdf/doc/pdftoppm.1

Gonzalez, R. C., & Woods, R. E. (2018). *Digital image processing* (4th ed.). Pearson. https://www.imageprocessingplace.com/DIP-4E/dip4e_main_page.htm

Gu, J., Goetschalckx, M., & McGinnis, L. F. (2010). Research on warehouse design and performance evaluation: A comprehensive review. *European Journal of Operational Research*, *203*(3), 539–549. https://doi.org/https://doi.org/10.1016/j.ejor.2009.07.031

Hancock, P. J., Baddeley, R. J., & Smith, L. S. (1992). The principal components of natural images. *Network: computation in neural systems*, *3*(1), 61.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2980–2988. https://doi.org/10.1109/ICCV.2017.322

Heidari, N., & Iosifidis, A. (2024). Geometric deep learning for computer-aided design: A survey. *arXiv preprint arXiv:2402.17695*.

Howard, R., & Björk, B.-C. (2007). Use of standards for cad layers in building. *Automation in Construction*, *16*(3), 290–297.

IEEE Global Initiative on Ethics of Autonomous and Intelligent Systems. (2019). Ethically aligned design: A vision for prioritizing human well-being with autonomous and intelligent systems [Accessed: 2025-05-09].

Intwala, A. (2020). Image to cad: Feature extraction and translation of raster image of cad drawing to dxf cad format. In N. Nain, S. K. Vipparthi, & B. Raman (Eds.), *Computer vision and image processing* (pp. 205–215). Springer Singapore.

Ivanov, D., & Dolgui, A. (2021). A digital supply chain twin for managing the disruption risks and resilience in the era of industry 4.0. *Production Planning & Control*, *32*(9), 775–788.

Jocher, G., Chaurasia, A., Qiu, J., & Stoken, A. (2023). YOLO by Ultralytics. github repository.

Khan, M. G., Huda, N. U., & Zaman, U. K. U. (2022). Smart warehouse management system: Architecture, real-time implementation and prototype design. *Machines*, *10*(2), 150.

Khanam, R., & Hussain, M. (2024). Yolov11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*.

Lee, H.-T., Cheon, H.-R., Lee, S.-H., Shim, M., & Hwang, h.-j. (2023). Risk of data leakage in estimating the diagnostic performance of a deep-learning-based computer-aided system for psychiatric disorders. *Scientific Reports*, *13*. https://doi.org/10.1038/s41598-023-43542-8

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2980–2988.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2117–2125.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *European conference on computer vision*, 740–755.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *Proceedings of the European Conference on Computer Vision (ECCV)*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2

Liu, W., Yang, T., Wang, Y., Yu, Q., & Zhang, L. (2024). Symbol as points: Panoptic symbol spotting via point-based representation. *arXiv preprint arXiv:2401.10556*.

Marcus, G. (2020). The next decade in ai: Four steps towards robust artificial intelligence. *arXiv preprint arXiv:2002.06177*.

Naumann, A., Hertlein, F., Dörr, L., Thoma, S., & Furmans, K. (2023). Literature review: Computer vision applications in transportation logistics and warehousing. *arXiv preprint arXiv:2304.06009*. https://doi.org/10.48550/arXiv.2304.06009

Niemistö, J. (2024). *Object detection in engineering diagrams with scarce training data* [Master's thesis, Aalto University] [Available at: https://aaltodoc.aalto.fi/items/37ac9688-7a21-42ed-91e0-b9d4acaf9bc2].

Niu, Z., Martin, R. R., Langbein, F. C., & Sabin, M. A. (2015). Rapidly finding cad features using database optimization. *Computer-Aided Design*, *69*, 35–50. https://doi.org/https://doi.org/10.1016/j.cad.2015.08.001

Nurminen, J. K., Rainio, K., Numminen, J.-P., Syrjänen, T., Paganus, N., & Honkoila, K. (2020). Object detection in design diagrams with machine learning. *Progress in Computer Recognition Systems 11*, 27–36.

Pujawan, N., Vanany, I., & Dewa, P. (2017). Human errors in warehouse operations: An improvement model. *International Journal of Logistics Systems and Management*, *27*, 298. https://doi.org/10.1504/IJLSM.2017.10005117

Qiu, H., Li, H., Wu, Q., Meng, F., Ngan, K., & Shi, H. (2019). A2rmnet: Adaptively aspect ratio multi-scale network for object detection in remote sensing images. *Remote Sensing*, *11*, 1594. https://doi.org/10.3390/rs11131594

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 91–99.

Rosebrock, A. (2015). Capturing mouse click events with python and opencv.

Sekachev, B., Manovich, N., Zhiltsov, M., Zhavoronkov, A., Kalinin, D., & Zharkov, A. (2020). Computer vision annotation tool (cvat) [GitHub repository].

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 1–48.

Smuha, N. A. (2019). The eu approach to ethics guidelines for trustworthy artificial intelligence. *Computer Law Review International*, *20*(4), 97–106.

Szeliski, R. (2010). *Computer vision: Algorithms and applications*. Springer.

Tong, K., & Wu, Y. (2024).
Small object detection using deep feature learning and feature fusion network.
Engineering Applications of Artificial Intelligence, *132*, 107931.
https://doi.org/https://doi.org/10.1016/j.engappai.2024.107931

Ultralytics. (2023). Yolov8 - official documentation [Accessed: 2025-03-18].

Ultralytics. (2024a). Yolov10 - object detection without nms [Accessed: 2025-03-20].

Ultralytics. (2024b). Yolov11 - next-generation real-time object detector [Accessed: 2025-03-20].

Van den Berg, J. P., & Zijm, W. H. (1999).
Models for warehouse management: Classification and examples.
International journal of production economics, *59*(1-3), 519–528.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features.
Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, *1*, I–I. https://doi.org/10.1109/CVPR.2001.990517

Wang, A., Chen, H., Liu, L., Chen, K., Lin, Z., Han, J., et al. (2024).
Yolov10: Real-time end-to-end object detection.
Advances in Neural Information Processing Systems, *37*, 107984–108011.

Wang, C., Zhao, Y., Li, H., & Lin, D. (2024).
Yolov9: Learning what you want to learn using programmable gradient information.
https://arxiv.org/abs/2402.13616

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q., & Kennedy, P. J. (2016).
Training deep neural networks on imbalanced data sets.
2016 International Joint Conference on Neural Networks (IJCNN), 4368–4374.
https://doi.org/10.1109/IJCNN.2016.7727770

Wang, S., Chen, C., Le, X., Xu, Q., Xu, L., Zhang, Y., & Yang, J. (2025). Cad-gpt: Synthesising cad construction sequence with spatial reasoning-enhanced multimodal llms.
Proceedings of the AAAI Conference on Artificial Intelligence, *39*(8), 7880–7888.

Wang, X., Zheng, J., Hu, Y., Zhu, H., Yu, Q., & Zhou, Z. (2024).
From 2d cad drawings to 3d parametric models: A vision-language approach.
arXiv preprint arXiv:2412.11892. https://arxiv.org/abs/2412.11892

Yang, L., Zhang, J., Li, H., Ren, L., Yang, C., Wang, J., & Shi, D. (2024).
A comprehensive end-to-end computer vision framework for restoration and recognition of low-quality engineering drawings. *Engineering Applications of Artificial Intelligence*, *133*, 108524.

Yin, H., Chen, C., Hao, C., & Huang, B. (2022).
A vision-based inventory method for stacked goods in stereoscopic warehouse.
Neural computing and applications, *34*(23), 20773–20790.

Yu, W., Shen, X., Hu, J., & Yin, D. (2021). Revisiting the loss weight adjustment in object detection.
arXiv preprint arXiv:2103.09488.

Zhang, H., Wang, Y., Dayoub, F., & Sünderhauf, N. (2021).
Varifocalnet: An iou-aware dense object detector.
2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 8510–8519.
https://doi.org/10.1109/CVPR46437.2021.00841

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., & Wang, O. (2018).
The unreasonable effectiveness of deep features as a perceptual metric.

2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 586–595.
https://doi.org/10.1109/CVPR.2018.00068

Zhang, S., Chen, J., Duan, Y., & Wang, X. (2022).
Dino: Detr with improved denoising anchor boxes for end-to-end object detection.
IEEE Transactions on Pattern Analysis and Machine Intelligence.

Zhang, W., Joseph, J., Yin, Y., Xie, L., Furuhata, T., Yamakawa, S., Shimada, K., & Kara, L. B. (2023).
Component segmentation of engineering drawings using graph convolutional networks.
Computers in Industry, *147*, 103885.

Zhang, X., Wang, Y., Li, M., & Zhao, L. (2023).
Superyolo: Super resolution assisted object detection in multimodal remote sensing imagery.
IEEE Transactions on Geoscience and Remote Sensing, *61*, 1–15.
https://doi.org/10.1109/TGRS.2023.3258666

Zhao, Y., Deng, X., & Lai, H. (2020). A deep learning-based method to detect components from scanned structural drawings for reconstructing 3d models. *Applied Sciences*, *10*(6).
https://doi.org/10.3390/app10062066

Zheng, Z., Li, J., Zhu, L., Li, H., Petzold, F., & Tan, P. (2022).
Gat-cadnet: Graph attention network for panoptic symbol spotting in cad drawings.
arXiv preprint arXiv:2201.00625.

Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., & Le, Q. V. (2020).
Rethinking pre-training and self-training.
Advances in Neural Information Processing Systems (NeurIPS).

Appendices

Appendix A Info about company

Consafe Logistics (CL) AB is a leading European supplier of Warehouse Management System, specializing in logistics software for optimizing supply chain performance. Headquartered in Lund, Sweden, the company provides digital solutions to streamline warehouse operations for clients across various industries. This thesis was conducted in collaboration with CL as part of an applied AI project focused on automating CAD-to-SQL warehouse layout conversion.