UPPSALA
UNIVERSITET

UPTEC STS 26011
Examensarbete 30 hp
Juni 2026

# Automated Recognition of Railway Signaling Components

## Using Machine Learning

Hanna Larsson
Julia Ploman

Civilingenjörsprogrammet i System i Teknik och Samhälle

UPPSALA
UNIVERSITET

# Automated Recognition of Railway Signaling Components Using Machine Learning

Larsson, Hanna
Ploman, Julia

## Abstract

This paper focuses on developing and evaluating machine learning methods to automate the extraction and structuring of information from railway signaling technical drawings. The goal is to address the challenge of digitizing paper-based documentation in the railway infrastructure sector. The study addresses this challenge by developing a pipeline that integrates object detection, classification, and optical character recognition (OCR) to identify and extract symbols and text from these technical drawings. The research focuses on two primary areas: symbols and text. For symbols, the project explores how machine learning-based object detection and classification can be designed to identify relay components. This involves a two-step approach: first, detecting the location of symbols using a YOLO (You Only Look Once) model, and second, classifying the detected symbols using deep learning architectures. The results show a very high performance in symbol detection, with near-perfect recall, and almost perfect accuracy in symbol classification, indicating that these symbols are easily learnable due to their consistent appearance. For text, the study tunes machine learning-based OCR methods to domain-specific data. This includes text detection using a YOLO model to locate text regions, and text recognition using a fine-tuned transformer-based OCR model to read the text content. While text detection performs strongly, text recognition, particularly the reading of specific character formats like Roman numerals and subscripts, presents the most errors in the overall pipeline. The results demonstrate that machine learning-based methods can effectively automate parts of the digitization process for railway signaling drawings. Individual components of the pipeline already perform at a level sufficient for practical use, and the work provides a strong foundation for further development toward a more complete automated system.

**Teknisk-naturvetenskapliga fakulteten**

**Uppsala universitet, Utgivningsort Uppsala**

Handledare: Martin Berg  Ämnesgranskare: Ewert Bengtsson

Examinator: Elísabet Andrésdottir

# Sammanfattning

I dagens digitaliserade samhälle finns fortfarande stora mängder kritisk teknisk information lagrad i form av pappersritningar. Detta är särskilt framträdande inom järnvägssektorn, där viktiga delar av infrastrukturen dokumenteras genom tekniska ritningar, ofta bevarade som fysiska original eller inskannade kopior. En del av denna dokumentation rör signalsystem, vars huvudsakliga funktion är att möjliggöra en säker tågtrafik på järnvägar och vid stationer. För signalsystemen utgör ritningar ett centralt underlag för drift och underhåll. Trots deras stora betydelse hanteras dokumentationen fortfarande till stor del manuellt, vilket gör både tolkning och hantering tids- och resurskrävande.

Mot denna bakgrund undersöker detta projekt hur maskininlärning kan användas för att automatisera digitalisering av signalritningar. Studien fokuserar på reläställverk modell 59, ett vanligt förekommande signalsystem där elektromagnetiska reläer styr signaler, växlar och spårledningar för att säkerställa trafiksäkerhet. Ritningarna som används för detta system består huvudsakligen av två kategorier: grafiska symboler och text. Symbolerna representerar olika reläkomponenter och förekommer i ett antal standardiserade varianter. Textinformationen omfattar bland annat relänamn, kontaktbeteckningar och kopplingsinformation som tillsammans beskriver systemets logik.

För att automatiskt extrahera information från dessa rutningar utvecklades en maskininlärningsbaserad process i flera steg. Först användes objektdetektion med en YOLO-modell (You Only Look Once) för att lokalisera symboler i ritningarna. Därefter klassificerades varje detekterad symbol med hjälp av en djupinlärningsmodell för att bestämma dess exakta symboltyp. Därefter användes ytterligare en YOLO-modell för att identifiera textregioner, som sedan bearbetades av en finjusterad, transformer-baserad OCR-modell (Optical Character Recognition) för optisk teckenigenkänning. Slutligen kopplades symboler och text samman genom en regelbaserad metod som utnyttjade deras relativa positioner i ritningarna. Den data som användes för att träna respektive modell bestod av manuellt annoterade ritningar från flera olika järnvägsanläggningar. Modellerna utvärderades på data som inte ingått i träningen för att bedöma hur väl de presterade på tidigare osedd information.

Resultaten från studien visade att den föreslagna metoden uppnådde hög prestanda i samtliga delsteg. Symboldetektionen uppvisade en mycket hög träffsäkerhet och klassificeringen av symboltyper nådde nära perfekt resultat. Textdetektionen presterade väl, medan textigenkänningen var den största utmaningen i systemet. Modellen var som mest felbenägen när texten på ritningarna innehöll romerska siffror, vertikal text och specialtecken som exponenter och nedsänkta symboler. Samtidigt förbättrades resultatet markant efter maskininlärningsbaserad justering till domänspecifik data.

I den slutgiltiga processen identifierades symboltyper korrekt i samtliga testfall, och strukturerade textattribut såsom kontaktnummer och namn uppnådde hög noggrannhet. Sammantaget uppnåddes cirka 75 % helt korrekta informationsrader, där majoriteten av felen härstammade från textigenkänningen. En del förbättringspotential identifierades också i kopplingen mellan symboler och tillhörande text, då den är känslig för variationer i ritningarnas layout, vilket gör att relationer ibland missas eller kopplas fel.

Trots begränsningar vad gäller textigenkänning och matchningslogik visar studiens resultat tydligt potentialen i att använda maskininlärning för digitalisering av teknisk järnvägsdokumentation. Den föreslagna metoden kan ligga till grund för automatiserad generering av reläkontaktlistor samt skapande av CAD-underlag. Genom att ersätta eller komplettera manuella arbetsmoment kan denna pipeline bidra till att minska tidsåtgång för dokumentation, reducera risk för mänskliga fel samt öka tillgängligheten till teknisk information. Detta är särskilt värdefullt i järnvägssektorn som i hög grad är beroende av historisk dokumentation.

Sammantaget visar studien att maskininlärningsbaserade metoder har tydlig potential för att förändra hur teknisk järnvägsdokumentation hanteras. Genom att kombinera olika maskininlärningstekniker i en sammanhängande process blir det möjligt att automatisera delar av ett annars mycket manuellt och tidskrävande arbete. Resultaten pekar på att detta kan bidra till mer effektiv informationshantering och skapa bättre förutsättningar för att tillgängliggöra och vidareutveckla äldre tekniska system i digital form.

# Acknowledgements

We would like to express our sincere gratitude to our supervisor Martin Berg and the team at Sweco for valuable support and insights throughout this project. We would also like to thank our subject reviewer Ewert Bengtsson at Uppsala University for important guidance and feedback during this thesis. Lastly, we would like to thank our examiner and program director Elísabet Andrésdóttir for long-term dedication and engagement throughout our studies.

# Table of Contents

# 1. Introduction

As more engineering processes become digital, the need for documentation that can be read and used digitally has increased. Despite this shift, large amounts of critical data across various engineering domains remain un-digitized or confined to paper-based records (Elyan et al., 2020, p. 91). This issue is especially clear in infrastructure-heavy industries such as the railway sector, which still heavily relies on traditional documentation methods. As a result, a lot of documentation is often maintained in paper-based or only partially digitized formats (Phusakulkajorn et al., 2023, p. 15).

An example of such documentation is engineering drawings, which are commonly used across different industries such as oil and gas, construction, mechanical engineering, and other engineering domains (Elyan et al., 2018, p. 1). Engineering drawings are especially difficult to process automatically, compared to other documentation. They typically contain a dense concentration of small symbols and highly detailed components, and large amounts of text (Jamieson et al., 2024, p. 2). A typical diagram may also include a wide variety of symbol types spanning multiple classes, often with only subtle visual differences between them. On top of this, these elements are often presented in inconsistent or non-standardized formats across different sources (Jamieson et al., 2024, p. 2).

This level of complexity makes the development of a fully automated system for interpreting, processing, and analyzing such drawings challenging (Elyan et al., 2018, p. 91). However, the demand for such a system is growing, as automated processing would enable more efficient and scalable management of engineering data. This is because engineering drawings often contain information with long-term value that must be preserved and made accessible. Also, recent advances in hardware, as well as in machine learning and computer vision methods, have significantly increased the feasibility of developing such systems (Elyan et al., 2020, p. 1).

These developments have increased interest in applying machine learning techniques to the interpretation of technical drawings. In particular, object detection, classification and optical character recognition (OCR) methods have shown potential for identifying symbols and retrieving information from scanned engineering documents (Jamieson et al., 2024; Mani et al., 2020). While previous research has explored the use of machine learning and computer vision for interpreting and digitalizing some types of engineering drawings, little to no research has been done specifically on railway signaling drawings. Railway signaling drawings are technical drawings used in safety-critical railway infrastructure (Trafikverket, 2023).

This challenge is relevant at Sweco, an engineering and architecture consultancy involved in planning and design of infrastructure systems, including railway signaling systems (Sweco, 2026a). At Sweco, a large part of the railway signaling documentation still exist mainly as scanned paper drawings varying in quality. The use of such documentation often requires manual handling and can limit efficient access to information. As the amount of legacy documentation remains large, there is an increased need for methods that can support the digitization of these drawings.

To address this problem, this thesis focuses on application of machine learning techniques for automated processing of railway signaling drawings. Using recent advances in object detection, classification, and optical character recognition (OCR), the study explores how different components and textual elements within such drawings can be automatically extracted and identified.

## 1.1 Aim and Research Questions

The aim of this thesis is to investigate whether modern machine learning methods for object detection, classification, and OCR can be used to automatically extract and structure information from railway signaling drawings. The project evaluates the performance and limitations of these methods when applied to technical drawings.

**RQ1:** How can machine learning-based object detection and classification be designed to identify and classify relay components in railway signaling drawings, and how well does the resulting approach perform?

**RQ2:** To what extent can machine learning-based OCR methods be adapted to extract text from railway signaling drawings?

# 2. Background

*This chapter presents the background of the thesis. It introduces Sweco as a company, describes railway signaling, and motivates the need for automation. The chapter also provides a brief technical background in machine learning and reviews relevant related research.*

## 2.1 Project Background

The project is carried out in collaboration with Sweco and is based on real challenges encountered in ongoing railway signaling projects.

### 2.1.1 Company Overview

Sweco is a European architecture and engineering consultancy that provides technical and advisory services in areas such as buildings and urban development, transportation infrastructure, water, energy and industry, and architecture. The company employs approximately 23,000 architects, engineers and other specialists who work on planning, design and development of infrastructure, buildings and urban environments. The company operates across Europe and is listed on Nasdaq Stockholm (Sweco, 2026a).

This thesis is written within Sweco's Transportation Infrastructure area. The area provides services in civil engineering, railways, traffic planning, project management, and IT (Sweco, 2026a). The project is conducted within the railways service area, with a specific focus on railway signaling.

### 2.1.2 Railway Signaling Systems

A railway signaling safety system enables vehicles to travel along a controlled train route without the risk of traffic accidents such as collisions or derailments (Trafikverket, 2023). Consequently, railway signaling systems play a crucial role in determining the capacity of the railway network and form the foundation for safe railway operations (Sweco, 2026b).

A central component of these systems is the railway interlocking system (ställverk), which controls and sets train routes through a track area in a way that prevents safety conflicts. Relay interlocking model 59 (reläställverk 59) is a relay-based railway interlocking system. It controls train routes by coordinating signals, switches and track circuits to ensure safe train movements and prevent conflicting routes. The system was designed for remote control and uses relay logic to manage route locking and safety functions (Trafikverket, 2020).

Railway signaling systems are documented in technical drawings that include symbols for relay components and associated textual information such as relay contact names and numbers.

### 2.1.3 Problem Description

A large part of the railway signaling documentation at Sweco consists of scanned paper drawings, which limits efficient reuse and digital processing of the information. This

creates a need for methods that can support the automated extraction of relevant information from such drawings.

If a model can be developed that automatically analyzes railway signaling drawings and extracts information such as symbol types, symbol locations, associated names, and contact numbers, this information can be used in several practical ways.

One potential application is the automatic creation of relay contact lists. By scanning a large set of signaling drawings from a specific site, it would be possible to generate a complete contact directory that shows all relay contacts for that site. Such relay contact lists already exist today, but they are created manually. They are widely used and considered valuable in daily engineering work.

Another important application is as a first step toward the digitization of paper-based signaling drawings. Even if the model cannot fully reconstruct the entire drawing structure, such as all connecting lines between symbols, the extracted symbols and related metadata can still be a useful digital foundation. For example, a list of symbols with coordinates and associated text could be imported into design software such as CAD (Computer-Aided Design) and used as a starting point. The system can therefore be seen as a supporting tool for digitization, rather than a fully automated replacement.

## 2.2 Technical Background

*Machine learning* (ML) is a part of artificial intelligence (AI) focused on algorithms that can learn patterns from training data and make inferences about new data. The ability to recognize patterns enables machine learning models to make decisions or predictions without hard-coded, explicit instructions (IBM, 2026a). There are two different machine learning approaches*: supervised learning* and *unsupervised learning.* Supervised learning uses labeled datasets to train algorithms, allowing them to identify patterns and improve predictive accuracy over time. Unsupervised learning, on the other hand, does not require any labeled data as it learns to uncover patterns and relationships on its own (IBM, 2026d). The training process is typically performed over multiple *epochs,* where each epoch represents one complete pass through the training dataset. A trained model's ability to generalize to unseen data is evaluated using *validation loss,* calculated on a separate validation dataset (Google, 2026).

*Computer vision* is a part of AI that uses machine learning to derive meaningful information from visual data such as images and videos. Within computer vision, *image recognition* refers to a system's ability to identify objects in an image, while *image classification* focuses on assigning images to predefined categories. *Object detection* is used to pinpoint where objects are in images, by drawing bounding boxes around them. Then, image classification categorizes the object (IBM, 2026c).

*Deep learning* is a subset of machine learning based on artificial neural networks. Inspired by the human brain, neural networks consist of interconnected layers of neurons, each of which performs a mathematical operation. This enables modeling of complex patterns and dependencies. Neural networks were introduced early in the history of machine learning but didn't gain widespread success until the late 2000s and early 2010s, due to advances in computational resources such as graphics processing units (GPUs) (IBM, 2026a).

*Convolutional neural networks* (CNNs) are a type of neural network commonly used for image classification and object recognition tasks. They operate on three-dimensional data and are well suited for computer vision applications. Compared to earlier approaches that relied on manual and time-consuming feature extraction, CNNs are a more scalable method for analyzing images by identifying patterns using ideas from linear algebra, such as matrix multiplication. However, training CNNs can be computationally demanding and often requires the use of GPUs (IBM, 2026b).

*Transformer models* are another neural network architecture. Originally developed to process sequential data and used in natural language processing, transformer models have also performed well in computer vision, where vision transformers (ViTs) can outperform CNNs on tasks such as image classification, object detection, and image segmentation. Transformer models work by using an attention mechanism to determine which information is most relevant (IBM, 2026e).

*Optical character recognition* (OCR), also known as text recognition, is a computer vision technique that extracts and converts text from images or scanned documents into a machine-readable format. Modern OCR uses machine learning techniques such as CNNs to train computers to read text inside images (Hasan, 2023; IBM, 2026c).

Together, these methods form the technical foundation for the object detection, classification, and OCR approaches discussed in the following research review.

## 2.3 Related Research

This section reviews relevant research related to image classification, object detection, OCR, and the digitization of technical drawings.

### 2.3.1 Image Classification Architectures

In 2012, Krizhevsky et al. published an article showing that large, deep convolutional neural networks can achieve record-breaking results on a challenging dataset called ImageNet using supervised learning. The authors introduced several important techniques, including the use of ReLU (rectified linear unit) activations to speed up training, GPU-optimized implementations to enable large-scale training, and methods to reduce overfitting despite the network's size. They used a deep convolutional neural network with five convolutional layers (Krizhevsky et al., 2012, p. 1-8).

In 2016, He et al. introduced Residual Networks (ResNet), a new neural network architecture. It was designed to address the degradation problem: as neural networks become very deep, they often become harder to train and may achieve worse accuracy than shallower models, despite having more layers. To address this problem, they proposed residual learning, where shortcut connections allow layers to learn residual functions instead of directly learning the full mapping. This makes training more stable and enables much deeper networks to achieve better performance. Using this approach, the authors successfully trained networks with up to 152 layers and achieved award winning results on the ImageNet dataset (He et al., 2016, p. 770-772).

More recently, research in image classification has moved beyond convolutional neural networks toward transformer-based architectures. Touvron et al. introduced Data-efficient Image Transformers (DeiT) in 2020 and showed that vision transformers

can be trained effectively on ImageNet alone, without convolutional layers, massive datasets, or large computing infrastructure (Touvron et al., 2020, pp. 1, 17-18).

### 2.3.2 Object Detection

One of the most widely used object detection frameworks is YOLO (You Only Look Once), a real-time object detection framework that was first launched in 2015 and later officially published in 2016 by Redmon et al. (2016). The software company Ultralytics has since developed and released several versions of the YOLO-algorithm, which are open-source and accessible to the public (Ultralytics, 2026).

A key characteristic of the YOLO model is that it treats object detection as one unified regression problem. Instead of breaking the object detection process into multiple steps, the entire image is analyzed all at once using a convolutional neural network (CNN). This single-stage design contrasts with earlier object detection approaches such as sliding-window classifiers and region-based methods. These methods typically use one stage to first generate potential object regions and then apply separate classifiers and post-processing steps to refine these detections. YOLO removes this separation by predicting locations of the objects and identifying their class labels simultaneously, which makes the detection process faster and more precise (Redmon et al., 2016, p. 1-2).

To analyze an image, YOLO first divides the image into an S x S grid. Each grid cell is responsible for detecting objects whose center lies inside it. For every grid cell, the model then predicts several bounding boxes, which are rectangular regions used to locate objects in the image. Each bounding box is described by its position, size and a *confidence score*. The confidence score represents how likely it is that an object is present in the box and also how accurately the box matches the object. In addition to this, each grid cell predicts class probabilities for the detected objects. During inference, these values are combined to produce a final confidence score for each object and class, indicating both the likelihood of the object and the quality of its localization (Redmon et al., 2016, p. 2). This process is visualized in Figure 1.

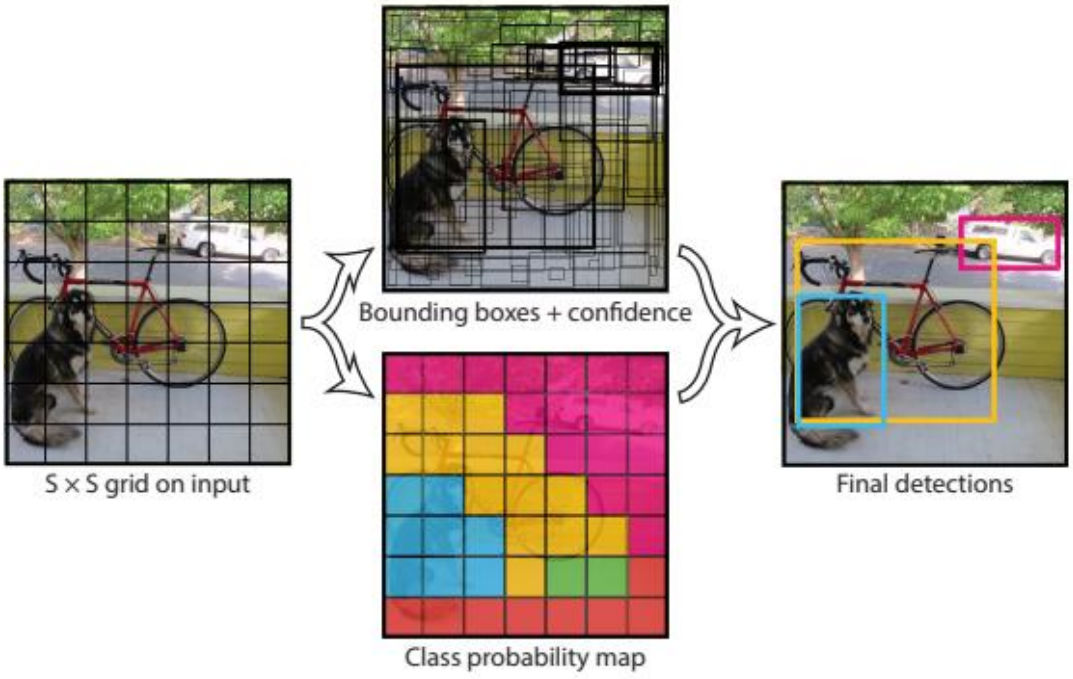



*Figure 1. The YOLO model (Redmon et. al, 2016, p. 2).*

### 2.3.3 Deep Learning for Digitization of Technical Drawings

Digitization of technical drawings has been researched long before the recent adoption of deep learning techniques (Mani et al., 2020, p. 674). In 2020, Mani et al. proposed a pipeline using a CNN classifier and a graph search approach to transform unstructured diagrams into structured information, which can enable a digital twin and applications such as machine learning based predictions (Mani et al., 2020, p. 673). Previous research has focused on domains such as architectural drawings, P&IDs (piping & instrumentation diagrams), floor plans, and other technical drawings (Jamieson et al., 2024, p. 6). The problem is usually divided into three elements: symbols, text and connectors, where specialized computer vision methods are needed to digitize each type (Jamieson et al., 2024, p. 5; Mani et al., 2020, p. 675).

Several challenges have been identified in the digitization of engineering diagrams and drawings. The diagrams are often complex, containing many similar and overlapping shapes. The presence of text is also a challenge, since it can appear anywhere in the diagram and in multiple fonts, scales, and orientations. Another challenge is the contextualization of extracted data, for example, relationships between text and symbols. Further challenges include low-quality drawings, imbalanced data, a lack of publicly available datasets, and the fact that annotation is needed for supervised learning algorithms, which is a time-consuming manual process (Jamieson et al., 2024, p. 3-4). Class imbalance, which means one or more classes are over-represented in a dataset, is a problem in both deep learning and traditional machine learning and often leads to poor performance on minority classes (Jamieson et al., 2024, p. 28).

For symbol detection and classification, models based on YOLO or CNNs are commonly used. In text digitization, there is a shift toward using deep-learning based methods (Jamieson et al., 2024, p. 10, 16). Connectors represent the relationship

between symbols, and can be represented by solid, dotted or dashed lines. It can be difficult for computer vision methods to distinguish between connectors and other shapes in the drawings, since all elements are composed of lines. Many line extraction approaches are still traditional, such as Hough transform- or kernel-based methods, and these can be sensitive to noise and line thickness. Jamieson et al. note that accurate connector detection from complex engineering diagrams remains difficult (2024, p. 20-21).

### 2.3.4 Optical Character Recognition

Technical drawings often contain large amounts of important textual information that are embedded into the graphical layout. This information is often transferred manually to other systems, making the process very time consuming and prone to human errors. By applying OCR, the textual elements in drawings can be automatically detected and converted into machine readable data, enabling further digital processing (Toro & Tarkian, 2025, p. 1-2).

Previous studies show that OCR usually performs the best compared to other methods when extracting text information from technical drawings. However, due to the complexity of these documents, many existing OCR solutions do not fully meet industrial requirements. An effective approach therefore needs to accurately detect and extract text while interpreting its meaning and identifying the information that is relevant for further processing (Toro & Tarkian, 2025, p. 2).

In OCR pipelines, the process is typically divided into two stages: text detection, which identifies where text appears in an image, and text recognition, which determines the content of the detected text regions (Toro & Tarkian, 2025, p. 2). Within technical drawings, OCR is often integrated with image processing methods that first segment the drawing into different elements before applying recognition algorithms to extract the relevant text (Toro & Tarkian, 2025, p. 1, 4). The use of OCR in this context supports the digitalization of engineering drawings and enables automated workflows in product design and manufacturing systems, such as design validation, quality control, and manufacturing analysis (Toro & Tarkian, 2025, p. 1, 17).

# 3. Methodology

*This chapter describes the methodology used in the thesis. It explains how the data was collected and prepared, how the machine learning pipeline was designed, and how the performance was evaluated.*

## 3.1 Data

This section describes the data used in the project and provides an overview of the drawings and their main components.

### 3.1.1 Data Collection

The data used in this project was provided by Sweco and consisted of technical drawings of railway signaling systems from multiple railway sites that all operate with relay interlocking model 59. Although each site included a wide range of drawing types, only those containing relay systems were selected, as they were relevant to the objectives of the project. An example of such a drawing is shown in Figure 2.

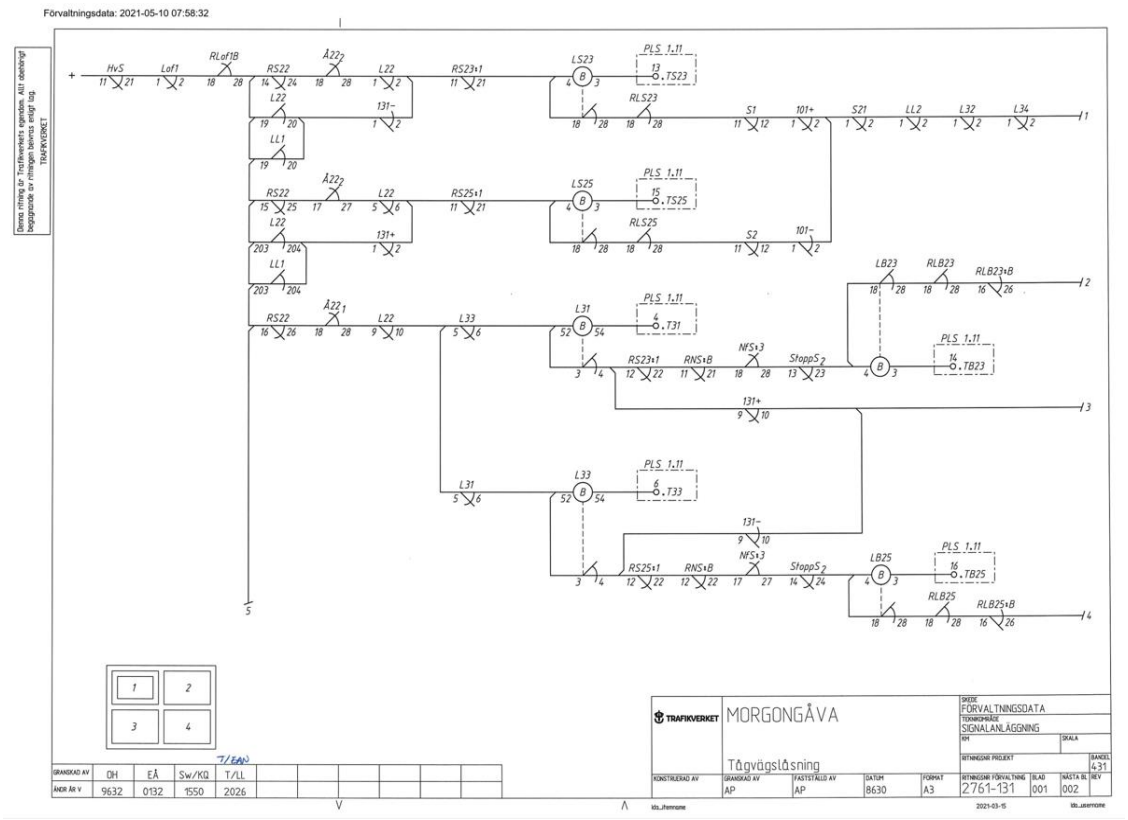



*Figure 2. Technical drawing of railway signaling system.*

The drawings are composed of different types of elements that together describe the functionality of the railway signaling system. These include standardized symbols representing relays, switches, and other electrical components, as well as lines that illustrate electrical connections. In addition, the drawings contain textual information such as labels, names, and technical annotations that provide further detail about each component and its role within the system. The combination of symbols, lines, and text

creates a structured and detailed representation of how the relay-based signaling system is designed and operates.

Among these elements, relay components play a central role in the drawings, as they form the foundation of the signaling logic and control functionality. A relay component consists of a coil (the electromagnetic actuator) and multiple relay contacts (switching elements). These symbols are the primary focus of this project. The relay contacts can be categorized into eight different types, representing all possible combinations of the following attributes:

- Type: *standard* or *industrial.*
- State: *engaged* or *disengaged.*
- Position: *front* or *back.*

These eight types are illustrated in Table 1. In practice, each relay contact may also appear in a mirrored orientation. Each of the eight types therefore also exists in a mirrored version, resulting in 16 different symbols for relay contacts in total**.**

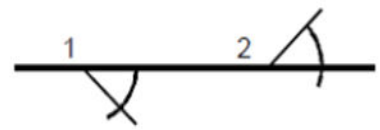
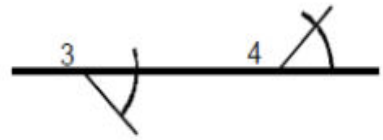
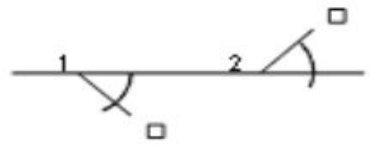
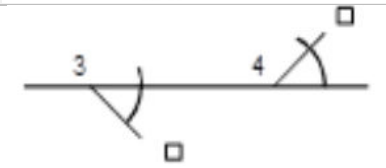
| Symbol | Type | Position | State |
|---|---|---|---|
|  | Standard relay contacts | 1. Front<br>2. Back | Engaged |
|  | Standard relay contacts | 3. Front<br>4. Back | Disengaged |
|  | Industrial relay contacts | 1. Front<br>2. Back | Engaged |
|  | Industrial relay contacts | 3. Front<br>4. Back | Disengaged |

*Table 1. Relay contacts (Trafikverket, 2025, p. 246-248).*

In addition to the relay contacts, there are also two kinds of relay coils, illustrated in Table 2:

- *Safety relay coil*,
- *Industrial relay coil*.

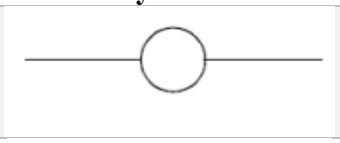
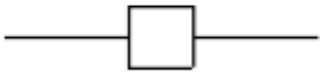
| Symbol | Type |
|---|---|
|  | Safety relay coil |
|  | Industrial relay coil |

*Table 2. Relay coils (Trafikverket, 2025, p. 246-248).*

In addition to graphical symbols, the drawings contain a significant amount of text that is essential for correctly interpreting the signaling system. One important category of text is relay identifiers, such as labels that uniquely identify each relay in the system. These identifiers indicate which relay coil and which individual relay contacts belong to the same physical relay, even when they appear in different parts of the drawing.

The drawings also include numerical annotations, referred to as contact numbers. These numbers identify specific physical contacts within a relay and are used to distinguish between multiple contacts of the same relay. Contact numbers make it possible to trace electrical connections across the drawings and ensure correct wiring.

In addition, the drawings contain higher-level descriptive text, such as titles and document information, including system name, location, or drawing number. Together, these textual elements complement the graphical symbols and provide the necessary context to understand the structure and function of the railway signaling system. An example of a relay representation is shown in Figure 3.

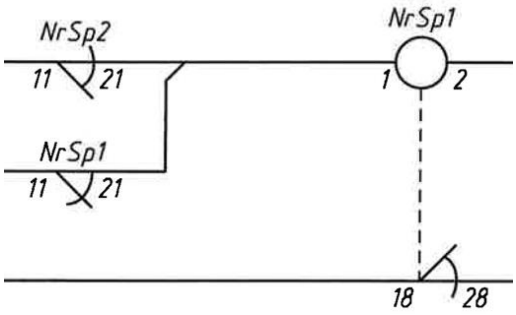



*Figure 3. Example illustrating how relays are represented in the drawings, including relay names and contact numbers.*

In Figure 3, the relay coil is labeled with the name “NrSp1” and associated contact numbers 1 and 2. Individual relay contacts appear as separate symbols and are labeled with relay names, indicating which physical relay they belong to. The dotted line indicates that the symbols share the same relay identifier, which is sometimes used to avoid repeating the relay name for symbols placed directly above one another. Lines in the drawings, including dotted lines, are not handled in this project. Instead, symbol-name associations are resolved in other ways.

### 3.1.2 Data Preparation

To enable supervised learning, the drawings were manually annotated to identify both symbols and text elements. The annotation process was carried out using Label Studio, an open-source data labeling platform (Label Studio, 2026). The platform enabled manual annotation of the technical drawings, where bounding boxes were drawn around each symbol of interest**.** Each annotated object was assigned a corresponding class label representing the type of relay symbol, such as relay contacts or relay coils, and the process was applied consistently across the dataset. An example of an annotated image is shown in Figure 4.

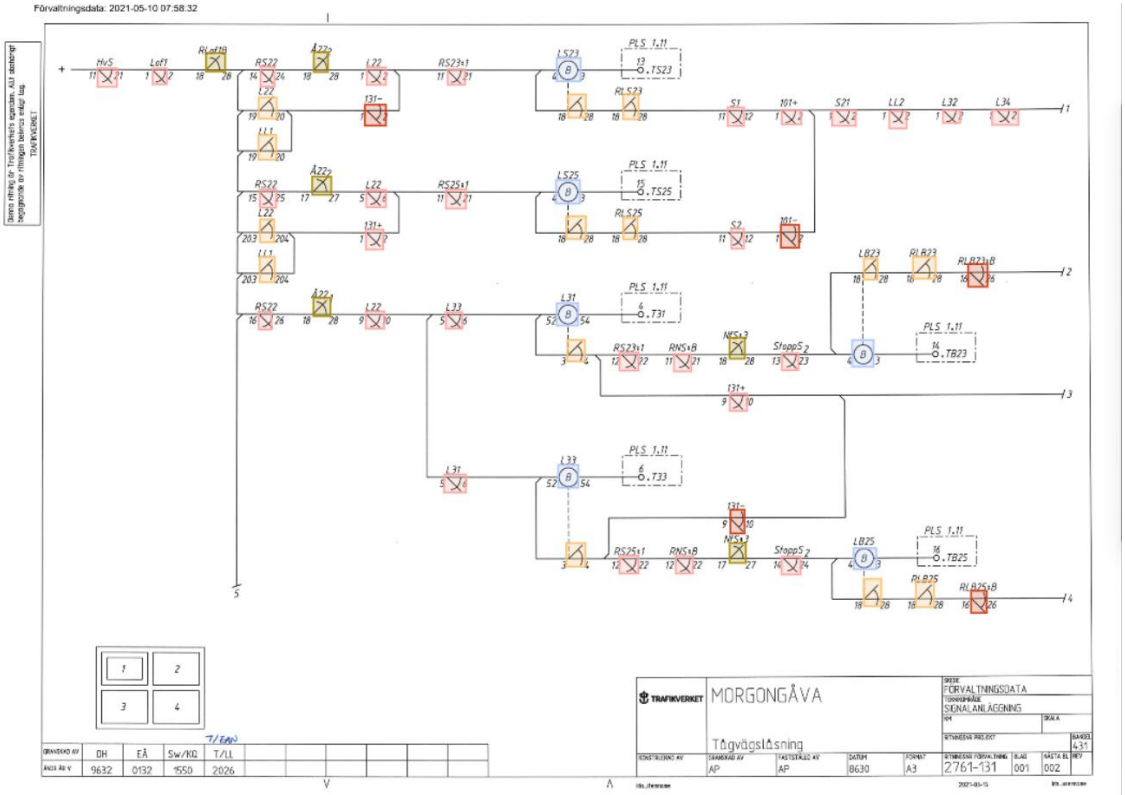



*Figure 4. Example of a technical drawing with annotated relay component symbols.*

A very similar procedure was applied to textual elements in the drawings. Bounding boxes were drawn around relevant text regions, and the content of each box was manually transcribed. An example of the text annotation is shown in Figure 5.

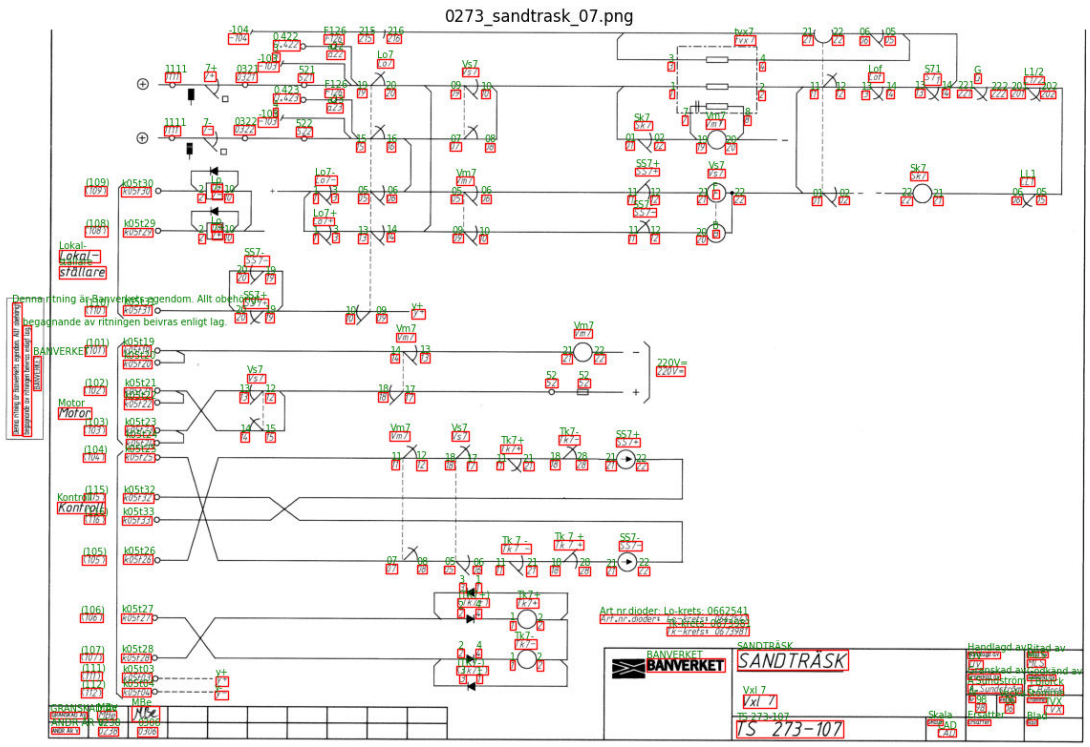



*Figure 5. Example of text annotation in Label Studio. Red boxes indicate annotated text regions, green strings show the corresponding ground-truth transcriptions.*

## 3.2 Proposed Approach

This project investigated how machine learning models can automate the process of finding and identifying symbols and associated text in technical drawings. To achieve this, a pipeline with five main stages was developed:

1) *Symbol Detection:* find where symbols are located in the drawing.
2) *Symbol Classification:* determine what each detected symbol represents.
3) *Text Detection:* find where text is located in the drawing.
4) *Text Recognition:* read the text inside each detected text box.
5) *Text-Symbol Linking:* assign the most likely label and contact numbers to each detected symbol.

The pipeline was implemented in Python and all models were trained on a Tesla T4 GPU.

## 3.3 Symbols

Symbol handling was divided into two steps: symbol detection and symbol classification. While some object detection models (such as YOLO) can both locate and classify objects into categories in a single step, a two-step visual approach for the symbols (detection followed by classification) was chosen over a single step for two main reasons:

*Preserving Details:* Processing large drawings at full resolution is computationally expensive, but resizing them to fit standard detection models can blur tiny details. In the

technical drawings, the relay symbols are often very small and visually similar, differing only in minor details. Standard YOLO models often lose these critical, pixel-level details when shrinking images, making it hard for them to tell small objects apart (Chandrashekhar et al., 2026). By cropping the detected symbols directly from the original high-resolution drawing and passing them to the classification model, the classifier receives a much clearer and more detailed image to analyze.

*Enabling Multi-Label Classification:* Standard detection models like YOLO assign objects to a single class, which struggles with the 16-class dataset for relay contacts. Separating classification into a second step makes it possible to use a custom "multi-head" model on the cropped images. This allows the model to predict visual attributes independently, solving the issue of having too little data for rare symbol combinations.

### 3.3.1 Symbol Dataset

A dedicated dataset was constructed to train and evaluate models for symbol detection and classification. In total, 85 technical drawings of railway signaling systems were labeled. The drawings varied in quality and level of detail, providing a representative sample of relay-based interlocking system drawings.

In these drawings, a total of 2642 symbols across 18 classes (16 relay contact variants and 2 coil variants) were annotated, forming a structured ground-truth dataset for training. The class distribution and corresponding Sweco CAD cell names are shown in Table 3.

| Nr | Symbol | Cell name | Instances |
|---|---|---|---|
| **1** | Back, Engaged | BD.V | 261 |
| **2** | Back, Engaged, Industrial | TBD.V | 29 |
| **3** | Back, Disengaged | BF.V | 233 |
| **4** | Back, Disengaged, Industrial | TBF.V | 12 |
| **5** | Front, Engaged | FD.V | 748 |
| **6** | Front, Engaged, Industrial | TFD.V | 31 |
| **7** | Front, Disengaged | FF.V | 421 |
| **8** | Front, Disengaged, Industrial | TFF.V | 128 |
| **9** | Mirrored, Back, Engaged | BD.H | 36 |
| **10** | Mirrored, Back, Engaged, Industrial | TBD.H | 3 |
| **11** | Mirrored, Back, Disengaged | BF.H | 21 |
| **12** | Mirrored, Back, Disengaged, Industrial | TBF.H | 11 |
| **13** | Mirrored, Front, Engaged | FD.H | 86 |
| **14** | Mirrored, Front, Engaged, Industrial | TFD.H | 2 |
| **16** | Mirrored, Front, Disengaged | FF.H | 73 |
| **16** | Mirrored, Front, Disengaged, Industrial | TFF.H | 14 |
| **17** | Industrial relay coil | TREL_X | 113 |
| **18** | Safety relay coil | RELX | 420 |

*Table 3. Class distribution of annotated objects in the dataset.*

Instead of treating the 16 relay-contact variants as separate classes, each relay contact is represented using four binary attributes: mirrored/not, industrial/standard, front/back,

and engaged/disengaged. This approach is called a multi-label or multi-head setup, and offers three main advantages:

1) *Better feature sharing:* The model learns visual traits across combinations (e.g., an "industrial" symbol has the same industrial feature whether it is front or back).
2) *Reduced class imbalance:* Evaluating attributes independently groups the data into much larger pools per decision. This overcomes the issue of rare combinations that only have 2 or 3 instances.
3) *Scalability:* Many relay-contact symbols share the same basic shape but can include additional markings that change their meaning. If every possible variant were treated as a separate class, the number of classes would grow very quickly as more variants are added. Using a multi-label (multi-head) approach makes it easier to add new features in the future without redesigning the entire class structure.

The resulting attribute distribution is shown in Table 4.

| Attribute | Count | Count |
|---|---|---|
| **Mirrored (Yes/No)** | 246 | 1,863 |
| **Industrial (Yes/No)** | 230 | 1,879 |
| **Position (Front/Back)** | 1,503 | 606 |
| **State (Engaged/Disengaged)** | 1,196 | 913 |

*Table 4. Attribute distribution for relay-contact symbols (derived from Table 3).*

### 3.3.2 Symbol Detection

Symbol detection focuses on locating relay components within the drawings. YOLO was selected for the symbol detection stage because of its unified, single-stage architecture, which efficiently processes entire images in one forward pass. As established in the related research section, YOLO-based models are well-suited for handling the complex layouts in engineering diagrams (Jamieson et al., 2024). This made YOLO a practical choice for locating the large number of small symbols present in railway signaling drawings.

Instead of evaluating the total number of training epochs, *early stopping* was used. This technique automatically stops the training process when the model's validation loss stops improving for a predefined number of consecutive epochs, which prevents overfitting and optimizes training time.

To determine the balance between detection and classification within the pipeline, three different class configurations of the YOLO model were trained and evaluated:

1) *1 Class (Detection only):* YOLO treats all symbols as a single generic "component" class. All classification is deferred to a separate model.
2) *3 Classes (Coarse classification):* YOLO distinguishes between three broad categories (safety coils, industrial coils, and relay contacts). Finer classification between relay contacts is handled by a separate model.
3) *18 Classes (Full classification):* YOLO does both detection and exact classification for all 18 specific symbol variations simultaneously.

All models were trained and evaluated using the identical dataset split (80% training and 20% validation). The main goal of this evaluation was to compare the results of the 1, 3, and 18-class models. If the simpler models (1 or 3 classes) were better at correctly locating the symbols, especially achieving a higher recall, than the 18-class model, it would prove that using a two-step pipeline was the right choice.

The YOLO11 version was selected because a recent benchmark study shows that the YOLO11 family performs well overall in both accuracy and efficiency compared to several earlier YOLO versions (Jegham et al., 2024). Two YOLO11 model sizes were tested: YOLO11s and YOLO11m. YOLO11s and YOLO11m use the same YOLO11 design but differ in size: YOLO11s is smaller and faster, while YOLO11m is larger and can be more accurate (Jegham et al., 2024). They were compared to evaluate the trade-off between smaller and larger YOLO models.

### 3.3.3 Symbol Classification

After detection, each symbol was classified to determine its type. Several deep learning architectures were tested and evaluated for the symbol classification task. All models were trained and evaluated under identical data preprocessing and training conditions to enable a fair comparison.

The input is a cropped image around one labeled symbol (i.e the bounding box). In the full pipeline, the classifier will receive bounding boxes from the YOLO detector, which may not be perfectly centered. To handle this, 10% padding is added around each box before cropping. This ensures the classifier can see the complete symbol even if the detection is slightly off-center, while keeping background noise small.

The classifier first decides which symbol type the crop contains. If the crop is classified as a relay contact, the classifier also predicts four additional binary attributes describing the relay contact (mirrored/not, industrial/standard, front/back, and engaged/disengaged):

1) *Symbol type* (3 classes): relay contact, safety coil, industrial coil.
2) *Relay-contact attributes* (only for relay contact): The model predicts four binary attributes using a multi-head setup. These attribute predictions are only meaningful for relay contacts.

The same 80/20 train/validation split used for detection is reused here for consistency. The split is done by drawings (not by individual crops). This means that all crops from a given drawing belong either to training or to validation. This reduces data leakage and avoids overly optimistic results.

Three different model architectures were compared: ResNet-18, EfficientNet-B0, and ConvNeXt-Tiny. All models use pre-trained ImageNet weights as initialization. Vision Transformers (e.g., DeiT) were considered but not prioritized, as transformer-based models typically require larger datasets to outperform CNNs on classification tasks (Dosovitskiy et al., 2021).

The three chosen architectures were selected to cover a range of model designs while remaining suitable for small datasets. ResNet-18 is a lightweight 18-layer residual network (Aicuflow, 2026) and serves as a baseline. EfficientNet-B0 aims to achieve

high performance with low computational cost through scaling strategies and has shown a good tradeoff between accuracy and computational complexity (Karataş & Atabas, 2026, p. 1-3). ConvNeXt-Tiny is a modern CNN architecture and has been shown to outperform other models on several image datasets (Jeevan & Sethi, 2024).

### 3.3.4 Evaluation Metrics

Several metrics were used to evaluate symbol detection and symbol classification.

Basic concepts used to measure accuracy in object detection are (Padilla et al., 2020, p. 238):

- *True Positive (TP):* A correct detection of a ground-truth bounding box.
- *False Positive (FP):* An incorrect detection, either of a non-existent object or a misplaced detection of an existing object.
- *False Negative (FN):* A ground-truth bounding box that was not detected.

To use these concepts, one needs to define what a "correct detection" is. A common definition is *Intersection over Union (IoU),* which measures the overlap between the predicted bounding box and the ground-truth bounding box. IoU is defined in Figure 6, where *area of overlap* is the intersection area between the predicted and ground-truth bounding boxes, while *area of union* is the total area covered by both boxes.

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} =$$

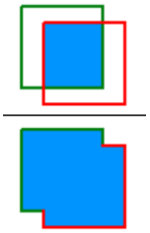


*Figure 6. Intersection over Union (IoU) (Padilla et al., 2020, p. 238).*

By comparing IoU with a threshold *t*, we can classify a detection as correct or incorrect (a detection is counted as a true positive if $IoU \geq t$). Using this, the assessment of object detection methods is mostly based on *Precision (P)* and *Recall (R)* (Padilla et al., 2020, p. 238).

*Precision* describes how many of the model's detections are correct, and is defined as

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections} \tag{1}$$

where *TP* is the number of true positives and *FP* is the number of false positives.

*Recall* describes how many of the ground-truth objects the model detects, and is defined as

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \tag{2}$$

where *FN* is the number of false negatives.

To summarize performance, object detection is often evaluated using *Average Precision (AP)* and *mean Average Precision (mAP)* (Padilla et al., 2020, p. 238-239). *AP* is recall levels as the confidence threshold is varied. AP can be interpreted as the area under the precision-recall curve for one class. *Mean Average Precision (mAP)* is then c
omputed as the average of AP over all classes, that is

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3}$$

where $AP_i$ is the Average Precision for class *i*, and *N* is the number of classes.

Because mAP depends on what is considered a correct detection, the IoU threshold must be stated. A common variant is mAP50, which uses a single IoU threshold of $t = 0.50$. Another common variant is mAP50-95, which averages AP over multiple IoU thresholds from 0.50 to 0.95 in steps of 0.05. Compared to mAP50, mAP50-95 is stricter because it rewards tighter, more accurate bounding boxes at higher IoU thresholds (Padilla et al., 2020, p. 238-241).

The symbol detection model will also be evaluated on training time because the project has limited computational resources and GPU access, so a model that achieves similar accuracy but trains faster is more practical to develop, tune, and retrain when the dataset or pipeline is updated.

Classification performance is also evaluated using Precision (1) and Recall (2), as well as their harmonic mean, the *F1-score*, defined as

$$F1 = 2\,\frac{P \times R}{P + R} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{4}$$

Additionally, overall *Accuracy* is reported, defined as the proportion of correct predictions:

$$Accuracy = \frac{correct\ predictions}{all\ predictions} \tag{5}$$

Because the validation set is imbalanced, where some classes are bigger than others, metrics are reported as weighted averages, where each class contribution is weighted by its frequency in the dataset. This makes the reported results reflect the symbol distribution in real drawings.

For the three-class symbol type prediction (relay contact, safety coil, industrial coil), accuracy, precision, recall, and F1-score are reported.

For the four binary attributes of relay contacts (mirrored, industrial, front/back position, engaged/disengaged state), each attribute is evaluated independently using accuracy, precision, recall, and F1-score. Metrics for each attribute are weighted averages across both binary classes (0 and 1).

### 3.3.5 Symbol Pipeline Evaluation

To evaluate symbol detection and classification performance on unseen data, a separate held-out test set from other railway sites was labeled and used for final evaluation. The full symbol pipeline (YOLO detection + symbol classification) was run on this dataset. This shows how well detection and classification work together on unseen drawings. The test set was not used during model selection or tuning.

30 drawings from two railway sites that were not in the training data were labeled. These drawings contained 176 safety coils, 4 industrial coils, and 1,045 relay contacts. The 1,045 relay contacts were distributed as shown in Table 5.

| Attribute | Count | Count |
|---|---|---|
| **Mirrored (Yes/No)** | 174 | 871 |
| **Industrial (Yes/No)** | 72 | 973 |
| **Position (Front/Back)** | 778 | 267 |
| **State (Engaged/Disengaged)** | 609 | 436 |

*Table 5. Distribution of attributes for relay contact symbols in the test dataset.*

Symbol pipeline performance was evaluated by comparing predicted bounding boxes to ground-truth boxes. A prediction was counted as a true positive if it matched a ground-truth symbol with $IoU \geq 0.40$. This lower IoU threshold was chosen because the symbols are very small and the ground-truth boxes are manually drawn, which means small annotation inconsistencies can have a large effect on the evaluation score.

## 3.4 OCR

OCR is treated as a separate component in the pipeline because text elements differ significantly from symbols in visual appearance, scale, and variability, and therefore require different modeling strategies. The OCR component of the pipeline is divided into two main steps: text detection and text recognition. The reason for these two steps is because there are no widely used models that can both detect and recognize text in a single step while also being fine-tunable for domain-specific data. End-to-end OCR solutions exist, but they typically operate as black-box systems and cannot be adapted to define what should be considered text or how domain-specific text should be interpreted.

### 3.4.1 OCR Dataset

The dataset for text detection and text recognition consisted of 70 drawings collected from 17 different sites. In total, 12,520 text bounding boxes were annotated, each paired with a corresponding transcription, forming a structured ground-truth dataset for training and evaluation of text recognition models. All text present in the drawings was annotated. This includes small numerical values and short labels located near symbols, as well as longer descriptive text such as titles and metadata in the drawing header. Annotating all text ensures that the OCR models learn to handle the full variation of text found in real railway signaling drawings.

A train-validation split was applied at site level, ensuring that all drawings from a given site were assigned to either the training or validation set. This strategy was used

consistently across all OCR processes to prevent data leakage and ensure a realistic evaluation setting. This was important because many text strings are repeated across drawings from the same site, meaning that a random split could lead to data leakage between the training and validation sets. This resulted in 9,305 training samples (74% of total) and 3,215 validation samples (26% of total).

### 3.4.2 Text Detection

For the text detection stage, YOLO11s was used to locate text regions within the drawings by predicting bounding boxes around individual text elements. The model was chosen for consistency with the symbol detection stage in the project. It was also chosen to keep training time low, since larger variants (e.g., YOLO11m) would take much longer to train on this dataset size.

All text boxes were trained as one class, “text”, with the goal of detecting all text in the drawing. Training was performed with an image size of 1,024, and early stopping was used.

### 3.4.3 Text Recognition

The text recognition stage focused on reading the content within the text bounding boxes. For this task, TrOCR was used, which is a transformer-based OCR model that can be fine-tuned with labeled datasets (Li et al., 2021, p. 1-2). During the selection process, PaddleOCR was also considered. PaddleOCR lightweight, open-source OCR framework that combines detection and recognition in a single pipeline (PaddleOCR, 2026). However, PaddleOCR was not chosen, partly due to its more complex setup and documentation.

TrOCR has been shown to outperform many traditional OCR models while remaining easy to implement and use (Li et al., 2021, p. 1-2). TrOCR was selected because it provides a strong pretrained baseline for reading text while still allowing fine-tuning on custom datasets. This makes it possible to adapt the model to the specific fonts, spacing, and formatting used in railway signaling drawings without training a text recognition model from scratch.

For text recognition, image crops were generated from the ground-truth annotated text bounding boxes and expanded by 10% padding on each side to include surrounding context. TrOCR was trained and evaluated on these ground-truth crops, meaning that text recognition was evaluated independently of the text detection stage.

To avoid data leakage, the dataset was split by railway site, so that text samples from the same site did not appear in both training and validation. Due to computational constraints, the validation set used for evaluation was randomly subsampled to 50% of its original size after the site-based split (1,607 samples from 3 sites).

Two fine-tuning settings were evaluated: (1) fine-tuning TrOCR on a 50% subsample of the training set (4,651 samples), and (2) fine-tuning on the full training set (9,303 samples). Both fine-tuned models were compared against the pretrained baseline model using the same validation set. This allowed the study to investigate both the benefit of fine-tuning and the benefit of increased training data, and to evaluate whether using the

full dataset provided a meaningful improvement over a smaller and less costly training set.

### 3.4.4 Evaluation Metrics

To evaluate the performance of the models detecting and recognizing text, several evaluation metrics were used.

*YOLO text detection performance* is reported using Precision and Recall (Eq. (1) and (2)) and summarized using mean Average Precision (mAP) (Eq. (3)), as defined in Section 3.3.4. Both mAP50 (using IoU threshold 0.50) and mAP50-95 (averaging IoU thresholds from 0.50 to 0.95) are reported from the Ultralytics validation output. In addition to accuracy metrics, training time is reported for the text detection model to show computational efficiency.

*Text recognition performance* is reported using *Exact Match Accuracy (EMA)* and *Character Error Rate (CER).*

*EMA* is the proportion of samples where the predicted transcription matches the ground truth exactly (case, punctuation, and spacing). This is a strict metric: one wrong character makes the whole sample incorrect (WWWInsights, 2024).

*CER* is the normalized Levenshtein distance between prediction and ground truth at the character level (Leung, 2021):

$$CER = \frac{S + D + I}{N} \tag{6}$$

where S is number of substitutions, D is number of deletions, I is number of insertions and N is number of characters in the reference text (ground truth). The output of the CER equation shows the ratio of characters in the ground truth that was incorrectly predicted in the OCR output. The lower the CER value (with 0 being a perfect score), the better the performance of the OCR model (Leung, 2021).

## 3.5 End-to-end Pipeline

The overall goal of the project is to automatically extract a structured component list from each railway signaling drawing. This means producing (for every drawing) a list of detected symbols where each symbol has the correct Sweco CAD cell name (symbol type/class), coordinates and the correct associated text information (symbol name and pair of contact numbers). Since symbols and text are predicted by separate models in the pipeline, an additional step is needed to connect each symbol to the relevant text located near it in the drawing.

As described in the pipeline overview, the full workflow consists of symbol detection and classification, followed by text detection and text recognition. Symbol classification outputs either a coil class or a relay-contact cell name. OCR outputs a set of text boxes with transcriptions. Finally, the end-to-end pipeline links these two outputs by assigning the most likely name text and contact-number texts to each detected symbol based on relationships in the drawing.

### 3.5.1 Connection Logic

The connection between symbols and text is done using a rule-based matching approach. After the symbol models have produced a set of symbol bounding boxes (with predicted symbol class/cell name) and the OCR models have produced a set of text bounding boxes (with strings), each symbol is processed independently.

For a given symbol bounding box, the algorithm defines expected "search regions" around the symbol where different types of text are likely to appear. They are:

- A *name region* located mainly above the symbol, where the symbol name/label is expected.
- *Number regions* around the symbol (below and/or to the sides), where contact numbers are expected.

Each OCR text box is scored as a potential match based on position relative to the symbol (distance and whether it lies in the expected region) and string type, meaning whether the recognized text looks "name-like" (mostly letters) or "number-like" (mostly digits). The highest scoring candidates are selected as the symbol's assigned name, left contact number, and right contact number, provided the best score exceeds a minimum threshold.

This step outputs a table with one row per symbol, including its bounding box coordinates, predicted cell name, and the matched text (name and left/right contact numbers). This is the main final result: a symbol list for each drawing with the text needed to interpret each symbol. As this step is rule-based rather than based on machine learning, it is not evaluated, as it lies outside the scope of this project.

### 3.5.2 End-to-end Evaluation

The full pipeline was evaluated on four drawings that were not included in either the symbol-model datasets or the OCR-model datasets. In total, these drawings contained 180 symbols.

For each drawing, a manual ground-truth list was created. One row is added for each symbol, and contains:

- Symbol type (CAD cell name),
- Symbol name/label,
- Left contact number,
- Right contact number.

The drawings were then processed by the full pipeline, and the predicted symbol lists were compared to the ground truth to identify which symbols and text assignments were correct and which error types occurred.

The following metrics are reported:

- Exact match: A row is counted as correct only if symbol type, name, left number and right number are all correct at the same time.
- Field accuracies: Accuracy is also reported separately for:
    - Symbol type,

- Name,
- Left contact number,
- Right contact number.

All end-to-end metrics are reported as averages across the four drawings, weighted by the number of symbols per drawing. For incorrect predictions, manual inspection is performed to determine whether errors in names or contact numbers are caused by missing text detection, incorrect text recognition, or incorrect matching logic between symbol and text.

# 4. Results

*This chapter presents the results of the developed pipeline, following the structure of the methodology chapter.*

## 4.1 Symbols

This section presents the results for the symbol-related parts of the pipeline, including symbol detection, symbol classification, and the combined symbol pipeline evaluation.

### 4.1.1 Symbol Detection

Symbol detection performance was evaluated with respect to variations in class configurations and model size, analyzing their effect on performance. All YOLO models were trained with early stopping, with a maximum of 250 epochs and a patience of 60 epochs. Training stopped if the validation loss did not improve for 60 epochs.

To evaluate how the number of detection classes affects performance, three models were trained and compared: a 1-class, 3-class, and 18-class model. To make the comparison fair, all three models used the same dataset split (80% training, 20% validation), the same model size (YOLO11s) and the same input image size (1024 pixels). This means that any performance differences mainly come from the class configuration, not from other training settings. The models were trained on 68 images (containing 2,116 symbols) and evaluated on 17 images (containing 526 symbols).

Table 6 shows the results for the three class configurations. Precision, recall, and mAP are reported across all classes.

| | **Best epoch** | **Training time [h]** | **Precision** | **Recall** | **mAP50** | **mAP50-95** |
|---|---|---|---|---|---|---|
| **1 class** | 125 | 0.252 | 0.986 | 0.981 | 0.990 | 0.516 |
| **3 class** | 157 | 0.310 | 0.988 | 0.963 | 0.986 | 0.541 |
| **18 class** | 99 | 0.228 | 0.747 | 0.756 | 0.832 | 0.419 |

*Table 6. Results from YOLO training using YOLO11s for 1, 3, and 18 classes.*

The results in Table 6 show the differences between the three class configurations. The 1-class and 3-class models achieved high recall (over 96%) while the 18-class model had a recall of 0.756. For the 18-class setup, some classes are rare or missing in the validation set due to strong class imbalance in the dataset, where several symbol variants have very few total instances.

Among the evaluated models, the 1-class model for symbol detection was selected for the rest of the pipeline. On the chosen validation split, it achieved the highest recall (0.981) and the highest mAP50 (0.990). The 3-class model for symbol detection performed similarly, but with slightly lower recall (0.963).

After selecting the 1-class configuration for symbol detection, YOLO11m, a bigger model than YOLO11s, was also tested for the same 1-class detection task. The same dataset split and image size were used, making the results directly comparable. Table 7 compares YOLO11s vs YOLO11m for the chosen 1-class setup.

| | Best epoch | Training time [h] | Precision | Recall | mAP50 | mAP50-95 |
|---|---|---|---|---|---|---|
| **1 class (YOLO11s)** | 125 | 0.252 | 0.986 | 0.981 | 0.990 | 0.516 |
| **1 class (YOLO11m)** | 119 | 0.416 | 0.992 | 0.989 | 0.987 | 0.515 |

*Table 7. Results from YOLO training using YOLO11m for 1-class detection.*

As seen in Table 7, YOLO11m achieved higher recall and precision than YOLO11s for the same task and was therefore used in the pipeline.

### 4.1.2 Symbol Classification

The symbol classification models predict the symbol type and, for relay contacts, additional attributes. Three deep learning architectures were compared for classifying cropped symbols into three main types: relay contact, safety coil, or industrial coil. All models were trained on 2,116 symbols and evaluated on 526 validation symbols. The validation set contained 395 relay contacts, 95 industrial coils, and 36 safety coils.

All models were trained for up to 50 epochs with early stopping (patience = 5 epochs). Table 8 shows the overall classification performance using weighted average metrics, which account for the class imbalance in the validation set. Parameters mean model size (millions of trainable weights). Best epoch shows when validation performance peaked before early stopping. All three models achieved perfect classification results.

| Model | Parameters [M] | Best epoch | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|---|---|
| **ResNet-18** | 11.18 | 3 | 1.000 | 1.000 | 1.000 | 1.000 |
| **EfficientNet-B0** | 4.01 | 1 | 1.000 | 1.000 | 1.000 | 1.000 |
| **ConvNeXt-Tiny** | 27.82 | 2 | 1.000 | 1.000 | 1.000 | 1.000 |

*Table 8. Performance comparison of 3-class symbol type classifiers.*

For crops classified as relay contacts, a multi-head model predicts four binary attributes: mirrored (yes/no), industrial (yes/no), position (front/back), and state (engaged/disengaged). The same three architectures were evaluated on the 395 relay contact symbols in the validation set.

Table 9 shows the per-attribute performance. Precision, Recall, and F1-score are weighted averages across both classes (0 and 1) for each binary attribute.

| Model | Attribute | Best epoch | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|---|---|
| **Resnet-18** | Mirror | 2 | 1.000 | 1.000 | 1.000 | 1.000 |
| | Industrial | | 1.000 | 1.000 | 1.000 | 1.000 |
| | Front/Back | | 1.000 | 1.000 | 1.000 | 1.000 |
| | Engaged | | 0.997 | 0.997 | 0.997 | 0.997 |
| **EfficientNet-B0** | Mirror | 24 | 1.000 | 1.000 | 1.000 | 1.000 |
| | Industrial | | 1.000 | 1.000 | 1.000 | 1.000 |
| | Front/Back | | 1.000 | 1.000 | 1.000 | 1.000 |
| | Engaged | | 1.000 | 1.000 | 1.000 | 1.000 |
| **ConvNeXt-Tiny** | Mirror | 27 | 1.000 | 1.000 | 1.000 | 1.000 |
| | Industrial | | 1.000 | 1.000 | 1.000 | 1.000 |
| | Front/Back | | 1.000 | 1.000 | 1.000 | 1.000 |
| | Engaged | | 1.000 | 1.000 | 1.000 | 1.000 |

*Table 9. Per-attribute performance of multi-head relay contact classifier.*

Based on the results in Tables 8 and 9, EfficientNet-B0 was selected for the final pipeline. While all three models achieved equivalent high accuracy (F1 ≥ 0.997) for both classification tasks, EfficientNet-B0 is the smallest with only 4.01M parameters.

### 4.1.3 Symbol Pipeline Evaluation

After evaluating detection and classification separately, the complete symbol pipeline was evaluated on a held-out dataset. In this section, some errors are exemplified by images.

In the symbol evaluation, a detection was counted as correct when it matched a ground-truth symbol with $IoU \geq 0.40$. The held-out dataset contained 1,225 symbols, and YOLO produced 1,249 detections. Detection results are shown in Table 10.

| TP | FP | FN | Precision | Recall | F1-score |
|---|---|---|---|---|---|
| 1220 | 29 | 5 | 0.977 | 0.996 | 0.986 |

*Table 10. Detection results on the held-out dataset.*

This corresponds to 1,220 correctly detected symbol boxes, 5 missed symbols, and 29 false detections. Figure 7 shows illustrative examples of false positives. False positives include other symbols as well as document features such as lines and corners.

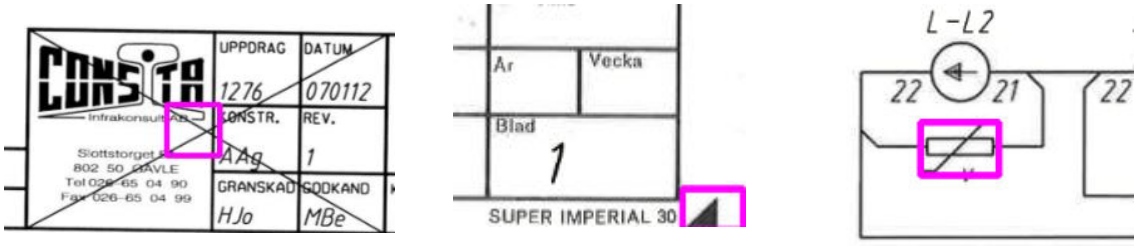



*Figure 7. Examples of false positives (FP) from three different drawings, shown in magenta boxes.*

Figure 8 shows examples of false negatives (missed detections). Missed detections occurred for example in lower-quality drawings or in areas where many symbols are close together. Out of 1,225 ground-truth symbols, 5 symbols were missed.

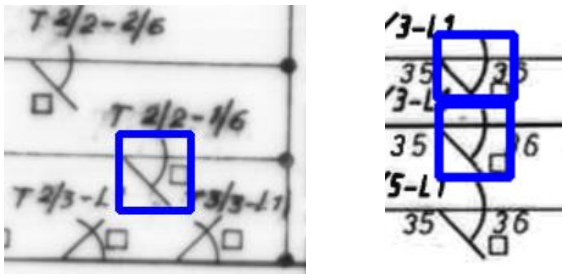


*Figure 8. Examples of missed detections from two different drawings, shown in blue boxes.*

For the 1,249 detections, the 3-class predictions are summarized in Table 11. The FP row shows how false detections were classified.

| GT / Predictions | Industrial Coil | Safety Coil | Relay Contact |
|---|---|---|---|
| **Industrial coil** | 4 | 0 | 0 |
| **Safety Coil** | 0 | 175 | 1 |
| **Relay Contact** | 0 | 0 | 1040 |
| **FP** | 3 | 0 | 26 |

*Table 11. Confusion matrix for 3-class classification. Rows represent ground truth and columns represent predictions.*

1,219 out of 1,220 correctly detected symbols were correctly classified in the first classification step. 26 out of 29 false positives were assigned to Relay Contact.

Detections that were classified as Relay Contact (1040 detections) were then passed to the second model that predicts the relay-contact attributes. The results are shown in Table 12. Table 12 includes only relay-contact detections that could be matched to a real, labeled symbol in the test data ($IoU \geq 0.40$). False detections that do not match any labeled symbol are not included.

| | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|
| **Mirror** | 1.000 | 0.994 | 0.997 | 0.999 |
| **Industrial** | 0.962 | 0.987 | 0.974 | 0.996 |
| **Front/Back** | 1.000 | 0.999 | 0.999 | 0.999 |
| **Engaged** | 0.997 | 1.000 | 0.998 | 0.998 |

*Table 12. Relay attribute classification results.*

Table 12 shows that accuracy exceeds 99% for all attributes, with Industrial achieving a slightly lower F1-score compared to the other attributes.

## 4.2 OCR

This section presents the results for the OCR component of the pipeline, including text detection and text recognition.

### 4.2.1 Text Detection

In Table 13, the results from the text detection model training are reported. All text boxes were trained as one class, “text”, with the goal of finding all text in the drawing. The model was validated on 19 drawings (3,215 text instances).

| Best epoch | Training time [h] | Precision | Recall | mAP50 | mAP50-95 |
|---|---|---|---|---|---|
| 78 | 0.862 | 0.948 | 0.926 | 0.956 | 0.621 |

*Table 13. YOLO text detection results.*

With precision 0.948 and recall 0.926, the results show that most predicted text boxes are correct and that the model detects most of the true text. Manual inspection showed that most false positives occurred when non-text graphics were detected as text, such as arrows inside other symbols and “+” signs that were not labeled as text. Some errors (both false positives and false negatives) were caused by differences in how long text strings were split into multiple boxes in the ground truth compared to the model’s predicted box segmentation, rather than truly missed detections. The few missed detections were mainly isolated single-digit labels (e.g., “2”, “3”, “4”) or occurred in areas where many small text boxes were close together. Examples of such text detection errors are illustrated in Figure 9.

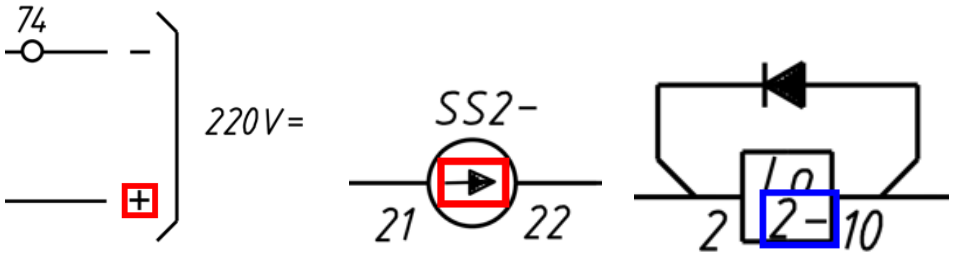



*Figure 9. Examples of text detection errors. False positives are shown in red boxes and false negatives (missed detections) in blue boxes.*

Manual inspection shows that the errors were largely limited to text outside the scope of the pipeline, as the scope is restricted to identifying relay symbol names and contact numbers. All examples shown in Figure 9 represent text types that are not used in subsequent pipeline steps.

### 4.2.2 Text Recognition

Text recognition was evaluated independently of text detection using ground-truth text crops and shows the effect of fine-tuning and training set size. The same validation set, consisting of 1,607 samples across 3 sites, was used for all comparisons.

Table 14 presents the validation results for the three model configurations: the pretrained TrOCR-base-handwritten baseline without fine-tuning, and two fine-tuned models trained for 3 epochs on 50% (4,651 samples) and 100% (9,303 samples) of the training set. The models are evaluated using Exact Match Accuracy, Character Error Rate (CER), and training time.

| Model | No of training samples | Training time [h] | Exact Match Accuracy | CER |
|---|---|---|---|---|
| **TrOCR baseline** | 0 | 0 | 0.221 | 0.612 |
| **TrOCR fine-tuned small** | 4,651 | 0.975 | 0.869 | 0.160 |
| **TrOCR fine-tuned full** | 9,303 | 1.833 | 0.878 | 0.130 |

*Table 14. TrOCR text recognition results on the subsampled validation set.*

Compared to the baseline model, both fine-tuned models achieved higher Exact Match Accuracy and lower CER on the validation set. Based on Table 14, the fine-tuned model trained on the full training set was selected for the final pipeline.

Manual inspection showed that the fine-tuned models predicted short labels and numbers correctly (e.g., “24”, “01”, “-172,1”), while errors were more common in long Swedish sentences.

## 4.3 End-to-end Pipeline

Finally, the full pipeline was evaluated on unseen drawings to assess overall performance in a realistic application setting.

For each drawing, the pipeline produced the same number of output rows as the number of ground-truth rows, since no false positives or false negatives were observed in the symbol detection stage in this small evaluation set. Table 15 summarizes the overall end-to-end results.

| Metric | Result |
|---|---|
| **Exact match** | 0.750 |
| **Symbol type accuracy** | 1.000 |
| **Name/label accuracy** | 0.772 |
| **Left number accuracy** | 0.972 |
| **Right number accuracy** | 0.978 |

*Table 15. End-to-end results (180 symbols).*

Table 15 shows that the predicted symbol type (CAD cell name) was correct for all symbols in the evaluation set. The name/label has the lowest accuracy of 77%, while the contact numbers are correct in about 97% of cases. Since an exact match requires all fields to be correct simultaneously, most exact-match errors are caused by incorrect name/label predictions.

During manual inspection of incorrect cases, recurring errors were observed in the recognized names/labels. Common examples were:

- Superscript or subscript (raised or sunken digits) being missed.
- Roman numerals in names being confused or removed.

An example of the two error types is shown in Figure 10.


*Figure 10. Example of an OCR error involving Roman numerals and subscript digits.*

In the example shown in Figure 10, the ground-truth name is “RSIIA2”, but the model predicts “RSIA”. This causes the name/label to be counted as incorrect, and therefore the row is not an exact match.

In addition, a small number of errors were related to the matching step rather than to text detection or text recognition. For example, contact numbers were assigned even when the corresponding ground-truth entry had no contact number. An example is shown in Figure 11.

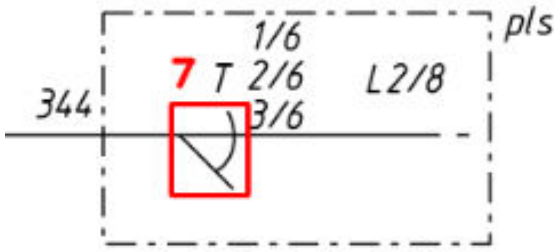



*Figure 11. Example where the pipeline assigns contact numbers even though none exist in the ground truth.*

In Figure 11, the ground truth for the symbol (index 7) does not contain any name/label or contact number, but the pipeline still assigns a name/label and contact numbers, leading to an incorrect match.

In addition to the error cases discussed above, the pipeline frequently produces fully correct outputs. Figure 12 shows an example region where all symbols are correctly detected and classified, all text elements are correctly recognized, and all associations between symbols and text are correctly matched.

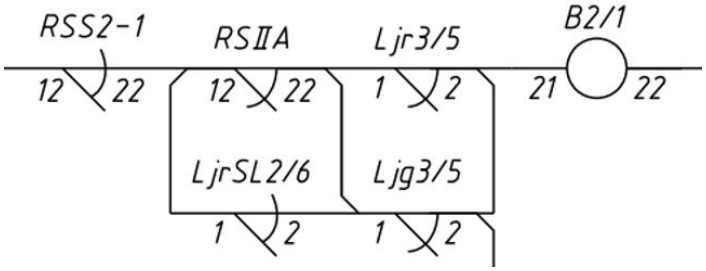



*Figure 12. Example region where all symbols and associated text are correctly detected, recognized, and matched by the end-to-end pipeline.*

# 5. Discussion

*This chapter discusses the results presented in the previous chapter by interpreting their meaning and relevance. The findings are evaluated in relation to the research objectives. In addition, the chapter addresses limitations of the study and discusses the implications of the findings.*

## 5.1 Symbols

This section discusses the results related to symbol processing, including symbol detection, symbol classification, and the implications of these results for the overall pipeline.

### 5.1.1 Symbol Detection

The symbol detection results demonstrate a very strong performance, especially for the 1-class configuration. The recall of 0.996 (see Table 10) in the symbol pipeline evaluation indicates that nearly all symbols were detected correctly, which is critical since only the detected symbols are passed on to the next stages.

An important choice in the evaluation was to use an Intersection over Union (IoU) threshold of 0.4 instead of 0.5, as is the standard when using Ultralytics YOLO. This decision was motivated by the nature of the data, where the relay symbols often are very small and manual ground-truth annotations may contain variations in bounding box placement and size. A higher threshold would be more sensitive to misalignments, even when the symbol is correctly localized. Therefore, performance is improved under the chosen, lower threshold, as slightly misaligned but otherwise correct detections are still counted as valid. Small differences in box tightness (mAP50-95) are less important than detecting the symbol at all, since the next steps can tolerate slightly imperfect boxes. Because the dataset is limited and the results are based on a single train/validation split, the measured differences between the 1-class and 3-class setups may vary with a different split.

The comparison between different class configurations further highlights the impact of dataset characteristics. While the 1-class and 3-class models achieved high performance, the 18-class model performed much worse. This can be explained by the class imbalance in the training dataset. Some relay types are more frequently occurring and therefore better represented, while others are less common and used less often. This leads to insufficient examples for the model to learn feature representations for minority classes, which results in a higher number of missed detections and misclassifications. This especially affects the 18-class model, which must detect and classify symbols in a single step, making it directly dependent on the class distribution shown in Table 3. Some classes contain as few as 2 or 3 instances, which is not sufficient to train or evaluate the model reliably.

### 5.1.2 Symbol Classification

The results from the symbol classification part of the project shows that all evaluated models achieved almost perfect performance, both for determining the symbol class and for predicting the associated attributes of relay contacts. This shows that all models

were capable to learn relevant features from the data, allowing them to accurately identify different symbol types as well as their associated attributes.

The extremely high results for the 3-class classification (see Table 8) can be explained by the nature of the data. The symbols originate from railway signaling drawings where the components follow a design standard and have minimal visual variation. They consist of simple black lines, often differing only in small details such as orientation, direction, or minor structural differences. Unlike real-world image classification tasks that often involve background noise, lighting variations, and varying shape and size, the images in the classification dataset are highly consistent in shape, size and structure. This reduces the complexity of the classification task, which explains the perfect results for the models.

The class imbalance in the dataset, especially the dominance of class relay contacts, seems to not have affected the results negatively. This shows the importance of using class weights during training and weighted metrics for evaluating the performance. This approach ensures that minority classes, like the safety coils, are appropriately represented both during the learning process and in the reported results.

However, the consistently perfect performance across Resnet-18, EfficientNet-B0 and ConvNeXt-Tiny for the 3-class classification indicates that the symbol dataset may be too simple to provide any meaningful insights into differences in model performance or capacity. When all evaluated models achieve identical and perfect results, it becomes difficult to determine which model actually offers real advantages when it comes to feature extraction, robustness, or learning ability. As a result, the evaluation of the models reflects the simplicity of the classification dataset rather than providing a reliable basis for comparing the actual strengths of the models. In more diverse datasets, differences between the architectures would likely become more distinct. Still, the comparison was conducted to ensure a consistent evaluation framework and also to confirm that all examined models were able to solve the task under the given conditions. At the same time, the perfect results show that the dataset was big enough for the models to learn the task, confirming that the labeled data was sufficient for this type of classification problem.

For the multi-head classification of relay contacts attributes, a similar pattern was observed (see Table 9). Most attributes achieved perfect scores across all models except for a small decrease in performance for the “engaged” attribute of the ResNet-18 model, 0.997 compared to 1.000. Nevertheless, the overall result confirms that a model predicting multiple attributes is very effective in this context.

Given that all three architectures achieved near equivalent performance, the model selection was primarily driven by efficiency considerations. EfficientNet-B0 has far fewer parameters than ResNet-18 and ConvNeXt-Tiny (4.01M compared to 11.18M and 27.82M, see Table 8) while still maintaining the same level of performance. Hence, it offers a lighter solution that still works just as well as the heavier models. The smaller size also offers an advantage when it comes to deployment in other applications, where computational resources and speed are important.

Although the reported classification results are extremely high, the reliability of the metrics varies between different symbol types and attributes. Some classes and attribute combinations are relatively rare in the dataset, which means that their performance

metrics are based on a small number of samples. For such cases, even a single misclassification can lead to noticeable changes in precision or recall, making the metrics less stable. This is particularly relevant for rare symbol types such as industrial relay coils, which occur only a few times, especially in the held-out symbol evaluation dataset. While the multi-head attribute-based method reduces the impact of class imbalance for relay contacts compared to treating all symbol variants as separate classes, performance results for the rarest cases should still be interpreted with caution.

There is a difference in classification performance when comparing the validation dataset with the held-out symbol dataset (Table 9 vs. Table 12). This can probably be explained by differences in how the symbol crops are obtained. In the validation set, the classification models are evaluated on manually annotated symbol crops, where it is guaranteed that all relevant visual information is present. In contrast, the held-out evaluation uses automatically detected symbol crops produced by the YOLO detection model. In these cases, parts of a symbol may be partially cropped or not fully visible, which can make classification more difficult. This effect is especially noticeable for the industrial relay contacts, which are identified by a small additional square symbol next to the main relay contact. In some automatically generated crops, the small square symbol is cropped out, leading to lower classification performance for the industrial attribute in the evaluation of the held-out set.

This behavior reflects a more realistic usage of the pipeline, where classification performance is influenced by the quality of the detection step. One possible improvement in future work could be to include more additional padding around the detected symbol crops before classification. However, increasing the crop size may also introduce more background noise, which could negatively affect classification performance. Therefore, such a trade-off would need to be evaluated.

## 5.2 OCR

This section discusses the results and limitations of the OCR models, including both text detection and text recognition, as well as their impact on the end-to-end pipeline.

### 5.2.1 Text Detection

The text detection achieved a strong performance, with a precision of 0.948 and a recall of 0.926 (see Table 13). This means that most text instances are detected correctly while still maintaining a relatively low rate of false positives. The drawings contain text under different conditions, including small annotations and text embedded in graphical structures. In this context, performing this well in both precision and recall reflects the robust ability of the YOLO-model to handle variability in how the text appears in the drawings, despite differences in font, size, and orientation.

Furthermore, the model achieves a high mAP50 score, which further confirms the detection quality. The value of 0.956 (see Table 13) means that 95.6% of the detected bounding boxes are considered correct at the chosen IoU threshold, indicating a very high overall detection and localization performance. A high detection quality is just as important in this pipeline as the symbol detection part, as accurate cropping directly affects the performance of the following recognition stage.

However, while the results show that while the model performs well overall, some challenges still remain. A closer examination of the errors reveals important limitations that are not fully captured by the metrics. Although the overall precision and mAP50 are high, indicating strong performance, they do not fully reflect specific failure cases.

One key issue is the occurrence of false positives in regions that contain graphical elements that can be mistaken for text visually. Arrows, line intersections and symbols like "+" often trigger detections by the model, since they share many features with regular text. Since these are not labeled as text in the dataset, the labeling policy affects the metrics. Detections that look like text or characters can still be counted as false positives if they are not part of the ground truth. It is worth noting that elements such as "+" could have been annotated as text, and doing so would not have been incorrect, it simply shows the choice of annotation standard within this project. Annotating them as text would likely have improved the text detection metrics, however, it would not affect the final output of the pipeline, as "+" characters are not used in the end result.

The false negatives, on the other hand, are primarily associated with very small or isolated text instances. Even though the overall number of such missed detections is very low, they do still occur in a few cases, almost exclusively involving single digits e.g. 1-9. They also only occur when these digits are present as a free-standing text instance, meaning the model can detect single digits when they appear as contact numbers next to symbols, but struggles with isolated single digits in other locations. This does not significantly impact the final result, however, as such cases are rare and do not carry critical information for this pipeline. Whole numbers are generally easier to detect since the presence of more digits provides a more distinct pattern that the model can identify more easily.

In addition, the digits are very small, consisting of very few pixels. Hence, it could also be a resolution limit related problem. Small characters may lose important visual details, making it harder for the model to predict their bounding boxes confidently. Therefore, it would be interesting to investigate whether using higher-resolution images throughout the pipeline could improve the detection of false negatives. Despite being small, these single digits can be crucial to understanding the drawing in other use case scenarios and therefore improving the detection of these is essential for future use of the models.

Something that also could improve the performance in text detection stage is using a larger YOLO-model, such as YOLO11m, with more parameters and greater learning capacity. However, these alternatives were not explored in this project due to computational and time related restraints. This reflection shows the trade-off between better performance and computational cost, which is very relevant in many machine learning projects. Larger models and longer training could improve detection accuracy, especially for small and more complex text instances, but they also require more computational power and longer training times, which can be limited in practice. Furthermore, larger models also have longer running times during inference, making them more expensive to use in production, which is an important factor to consider when deploying models in real-world applications.

### 5.2.2 Text Recognition

The results for text recognition improves after fine-tuning the pretrained TrOCR model using domain specific data from the railway signaling drawings. Both Exact Match Accuracy (0.878) as well as Character Recognition rate (0.130) significantly outperforms the baseline model (0.221 and 0.612 respectively), see Table 14. This is because fine-tuning adapted the TrOCR to the specific characteristics of the text in the dataset used. General OCR models, such as the baseline TrOCR, are trained on broad and diverse text sources and therefore lack specialization for technical drawings. Through the fine-tuning, the model learns these domain specific features, like inconsistent spacing or non-standard typography. It can therefore be concluded that domain adaptation is very important when applying general OCR models to technical drawings, in order to achieve a more reliable performance.

The results in Table 14 also indicate that training data set size has an effect on performance, as the model trained on the full dataset (9,303 samples) outperforms the model trained on half the data (4,651 samples) in both Exact Match Accuracy (0.878 vs. 0.869) and CER (0.130 vs. 0.160). Although the improvement is small, it suggests that increasing the amount of training data further could lead to better performance, particularly for the more complex text cases discussed below.

Since the model in the final evaluation, see Table 15, makes errors with specific characteristics like roman numbers, sub- and superscript, it can be concluded that some limitations still remain in handling more complex text that is found in technical drawings. The dataset used for evaluating text recognition is also very small and contains a lot of these tricky cases. This affects the evaluation, since a dataset with fewer edge cases would be easier for the model to handle. The issue with roman numerals, sub- and superscript could be handled by using a more advanced OCR model, or by further training the model specifically on these cases by increasing the amount of training data containing them. This could help the model better learn these structures and reduce this type of errors.

Furthermore, a better result could also have been achieved by using a different OCR model. A big challenge in this project was to find an OCR model that was both suitable for this context as well as possible to fine-tune with the data. This led to the selection of TrOCR, a transformer-based architecture that is adaptable to domain-specific data. However, there are other OCR-models that could have been used as well. For example, the more lightweight PaddleOCR. This would make it faster to run, which can be an advantage when handling large volumes of data. If PaddleOCR had been used and fine-tuned, there is a possibility that the results could look different, as it is a completely different kind of OCR. However, TrOCR still proved to be a suitable choice as it was relatively straightforward to use and produced stable results after fine-tuning.

## 5.3 Overall

This section describes the main limitations of the study, explains how the pipeline affects the results, and discusses possible applications and future improvements.

### 5.3.1 Limitations

One important limitation of this study is that no cross-validation was performed. Cross-validation is a common evaluation method where the dataset is split into several different training and validation sets, and the model is trained and evaluated multiple times. This helps reduce the dependence on a single data split and makes the results more robust.

In this project, all results are based on one fixed train/validation split. Since the drawings originate from different sites and vary in quality, font, and layout, the reported performance metrics could differ slightly if another split had been used. The datasets are also relatively small, as all data was manually labeled, which increases the sensitivity to how the data is divided. Cross-validation was not feasible due to long training times, limited computational resources, and time constraints within the project. Therefore, the reported metrics should be seen as indicative rather than definitive.

Another limitation is that the end-to-end pipeline evaluation was based on only four drawings, which is a very small sample. This was due to the time-consuming structure of the evaluation process, as contact lists had to be manually created for each drawing to be the ground truth for comparison. In addition, the current evaluation method made it difficult to see where in the pipeline errors occurred. For example, whether a mistake originated from text detection, text recognition, or the symbol-to-text linking step, as this required significant manual inspection. In future work, the end-to-end evaluation could be made both larger and more efficient, for example by using already existing contact lists as ground truth rather than creating them manually.

### 5.3.2 Pipeline and Application

Beyond evaluation uncertainty, some limitations are related to the structure of the pipeline itself. Some errors observed in the end-to-end evaluation are not caused by symbol detection, classification, or OCR, but by the rule-based logic used to link symbols with their associated text. This matching step relies on predefined rules and spatial relationships, which can fail in some edge cases.

This linking component was not evaluated separately, as it does not involve machine learning, but instead relies on manually designed rule-based logic, and was not the main focus of the project. However, its behavior directly affects the final output and performance of the pipeline. In future work, this part of the pipeline could be evaluated more systematically, for example by creating larger datasets with manually annotated symbols-to-text associations. Such data could be used either to evaluate the existing logic or to replace it with a machine learning-based approach.

The symbols-to-text matching could potentially have been improved by incorporating a predefined list of allowed label names and valid contact number pairs, since many labels appear to be recurring and there are well-defined rules for which number pairs belong together. By constraining the matching process to only recognized combinations, the system could reduce incorrect matches and hence get a higher Exact Match Accuracy. However, the full set of possible labels across all drawings is unknown, and a constrained list risks excluding valid labels that were not observed during development.

Since the pipeline consists of multiple sequential steps, errors can pass through the system. A small error in text recognition or symbol-text linking can cause the entire output row to be incorrect, even when all other components perform well. The exact-match metric is intentionally strict, as it requires all fields to be correct simultaneously. While this is useful for evaluation, it may underestimate the practical usefulness of the pipeline in scenarios where partial correctness is enough.

The practical usefulness of the pipeline depends on the intended application. For tasks such as symbol digitization or importing symbols into CAD systems, the very high performance of symbol detection and classification is sufficient and highly valuable. However, for applications such as generating contact lists or textual documentation, accurate text recognition becomes more critical. Even minor OCR errors in names/labels can significantly reduce usability, and this is where the model sometimes shows poor performance. This demonstrates that performance requirements and results must be considered in relation to the intended application.

### 5.3.3 Future Improvements and Generalization

Several architectural improvements could potentially have further increased performance but were not explored. One such improvement is the use of tiling strategies. Tiling means that a large image is divided into smaller overlapping parts, which are then processed separately by the detection model. This allows small objects to appear larger in each tile, making them easier to detect. A tiling approach could improve the performance of YOLO-based object detection, particularly for small symbols and small text.

Furthermore, larger models or longer training might have improved performance, especially for difficult cases. These options were not feasible within the available computational resources and project time frame. As a result, the presented solution represents a balance between performance and practicality rather than a fully optimized system.

The models were trained and evaluated only on one specific type of railway signaling drawings. As a result, the models are highly specialized to this type of data. If applied directly to other drawing standards, layouts, symbols, or fonts, the performance would likely degrade, since the models have not been exposed to such variations during training. This applies to all parts of the pipeline, including symbol detection, text detection and recognition, as well as the rule-based linking between symbols and text. The models have learned patterns, layouts, and conventions that are specific to this drawing type, which limits their ability to generalize to other domains.

However, while the trained models themselves are specialized, the overall pipeline and methodology are general. The same principles for data preparation, model training, and pipeline design could be applied to train new models on other types of drawings or standards, given appropriate annotated data.

# 6. Conclusions

*This chapter summarizes the key outcomes of the project and presents the overall conclusions drawn from the conducted work. It highlights the main contributions of the study and reflects on the extent to which the research objectives were fulfilled.*

## 6.1 Summary of Findings

This project investigated whether modern machine learning methods for object detection, classification, and OCR can be used to automatically extract information from railway signaling drawings. More specifically, it examines how machine learning models for detecting and classifying symbols can be designed and evaluated, and how an OCR approach can be adapted to extract textual information from the same type of drawings. Based on the results, this aim has largely been achieved, as the developed pipeline demonstrates that such methods can successfully be applied to this domain.

Symbol detection achieved a very high performance with almost perfect recall, ensuring that most symbols on the drawing are found. Symbol classification also performed very well, with almost all evaluated models reaching perfect results. This leads to the conclusion that the symbols in railway signaling drawings are highly learnable due to their consistent appearance throughout the dataset.

The OCR, on the other hand, showed a more varied performance. Text detection performed strongly, successfully identifying most relevant text regions, while text recognition remained the main source of errors in the final pipeline. It improved significantly using fine-tuning, but still struggled with some character formats like Roman numerals, subscript and superscript.

Despite the limitations, this project demonstrates that it is entirely possible to automate important parts of the process of interpreting and digitizing railway signaling drawings. This indicates a strong potential to reduce manual work, save time, and improve consistency in handling technical documentation. Even though the full pipeline is not perfect, individual components could be integrated as supportive tools in existing workflows. The results therefore demonstrates the practical potential of machine learning based approaches in the context of digitalizing paper-based drawings, while also pointing to clear opportunities for further development toward a more robust and fully automated system**.** This study provides a foundation for further exploration in this area, and can serve as a starting point for developing automated solutions for railway documentation. This is a promising and largely unexplored direction, with great potential for improvement and refinement.

Overall, the results demonstrate that machine learning based methods for object detection, classification, and OCR can effectively automate large parts of the digitization process, which was the purpose of the study.

## 6.2 Future Work

The results of this project open several directions for continued research and development.

Future work could focus on detecting the lines that connect symbols in the drawings. These lines represent relationships between components and are important for understanding the overall structure of the system. By using image analysis models for line detection and graph construction, symbol and text detection could be combined with connection logic, resulting in a more complete approach to drawing digitalization.

This would include representing the output of the pipeline as a structured graph, where symbols are nodes and connections are edges. Such a representation would make it easier to understand how different parts of the system are related, and to store the extracted information in a structured format that can be used in other tools or systems.

However, detecting lines in technical drawings also has challenges. Drawings often contain many different types of lines, not all of which are part of the actual system logic. Examples include lines that are part of symbols, borders in the drawing frame, or lines introduced during scanning, such as folds or noise in the paper. Distinguishing meaningful connections from irrelevant lines is therefore important.

As a result, a line detection component is unlikely to be fully accurate in all cases. While it is likely possible to develop models that perform well on average, the risk of incorrect or missing connections remains, which can be critical for the intended application. For this reason, such functionality should be viewed as an assistive tool rather than a fully automatic solution, where a human user can review, validate, and adjust the detected connections when necessary.

A challenge for future development is the need for domain knowledge. Developing a tool that fully understands the logic and structure of railway signaling systems requires a deep understanding of the domain. Without this knowledge, it is difficult to define rules, validate outputs, or make design choices about how the system should interpret the drawings. For a reliable solution to be developed, close collaboration with domain experts would be needed.

Another important direction for future work is the development of a human-machine interface for practical deployment. Such an interface should allow users to easily review the extracted information and correct or remove incorrect detections. The system could also highlight uncertain or unusual predictions, based on model confidence scores or simple logical rules.

# References


Aicuflow (2026). ResNet-18. Available at: https://www.aicuflow.com/docs/tool/training/computer_vision/image_classification/resnet-18 (Accessed: 19 May 2026).

Chandrashekhar, A., Satyanarayana, B., Gorrepati, R. R., et al. (2026). An efficient YOLOv12-based framework for detecting extremely small-scale objects. *Sci Rep*. [Online] 16, 2062.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale, *arXiv preprint arXiv:2010.11929*. [Online].

Elyan, E., Garcia, C. M., & Jayne, C. (2018). Symbols Classification in Engineering Drawings. *2018 International Joint Conference on Neural Networks* (IJCNN), 1–8.

Elyan, E., Jamieson, L., & Ali-Gombe, A. (2020) Deep learning for symbols detection and classification in engineering drawings. *Neural networks.* [Online] 12991–102.

Google (2026). Machine Learning Glossary. [Online]. Available at: https://developers.google.com/machine-learning/glossary (Accessed: 19 May 2026).

Hasan, M. (2023). OCR with Deep Learning in PyTorch (EasyOCR). Medium. Available at: https://eng-mhasan.medium.com/ocr-with-deep-learning-in-python-e443970d09e4 (Accessed: 19 February 2026).

He, K., Zhang, X., Ren, S., & Sun, J. (2016) Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). [Online].

IBM (2026a). What is machine learning? Available at: https://www.ibm.com/think/topics/machine-learning (Accessed: 11 February 2026).

IBM (2026b). Convolutional neural networks. Available at: https://www.ibm.com/think/topics/convolutional-neural-networks (Accessed: 11 February 2026).

IBM (2026c) What is computer vision? Available at: https://www.ibm.com/think/topics/computer-vision (Accessed: 12 February 2026).

IBM (2026d) Supervised versus unsupervised learning: What's the difference? Available at: https://www.ibm.com/think/topics/supervised-vs-unsupervised-learning (Accessed: 12 February 2026).

IBM (2026e). What is a transformer model? Available at: https://www.ibm.com/think/topics/transformer-model (Accessed: 12 February 2026).

Jamieson, L., Francisco Moreno-García, C., & Elyan, E. (2024). A review of deep learning methods for digitisation of complex documents and engineering diagrams. *The Artificial intelligence review*. [Online] 57 (6).

Jeevan P, P. & Sethi, A. (2024). Which Backbone to Use: A Resource-efficient Domain Specific Comparison for Computer Vision. *arXiv*. Available at: https://arxiv.org/abs/2406.05612 (Accessed: 19 May 2026).

Jegham, N., Koh, C. Y., Abdelatti, M., & Hendawi, A. (2024). Evaluating the Evolution of YOLO (You Only Look Once) Models: A Comprehensive Benchmark Study of YOLO11 and Its Predecessors. *arXiv*. Available at: https://arxiv.org/abs/2411.00201 (Accessed: 16 March 2026).

Karataş, H. & Atabas, İ. (2026). Image-Based Classification of Olive Varieties Native to Türkiye Using Multiple Deep Learning Architectures: Analysis of Performance, Complexity, and Generalization. *arXiv.* Available at: https://arxiv.org/pdf/2602.18530 (Accessed: 19 May 2026).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems, 25.*

Label Studio. (2026). Label Studio - Open Source Data Labeling Tool. [Online]. Available at: https://labelstud.io/ (Accessed: 23 March 2026).

Leung, K. (2021). Evaluate OCR Output Quality with Character Error Rate (CER) and Word Error Rate (WER). *Towards Data Science*. Available at: https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510/ (Accessed: 20 April 2026).

Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2021). *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models*. arXiv. Available at: https://arxiv.org/abs/2109.10282 (Accessed: 16 April 2026).

Mani, S., Haddad, M. A., Constantini, D., Douhard, W., Li, Q., & Poirier, L. (2020). 'Automatic Digitization of Engineering Diagrams using Deep Learning and Graph Search', in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition workshops*. [Online]. 2020 IEEE. pp. 673–679.

PaddleOCR (2026). PaddleOCR Documentation. Available at: https://www.paddleocr.ai/main/en/index.html (Accessed: 19 May 2026).

Padilla, R., Netto, S. L., & da Silva, E. A. B. (2020). A Survey on Performance Metrics for Object-Detection Algorithms. *2020 International Conference on Systems, Signals and Image Processing* (IWSSIP). pp. 237–242.

Phusakulkajorn, W., Núñez, A., Wang, H., Jamshidi, A., Zoeteman, A., Ripke, B., Dollevoet, R., De Schutter, B., & Li, Z. (2023) Artificial intelligence in railway infrastructure: current research, challenges, and future opportunities. *Intelligent transportation infrastructure*. [Online] 2.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [Online].

Sweco (2026a). About us. Available at: https://www.swecogroup.com/about-us/ (Accessed: 11 February 2026).

Sweco (2026b). Trafikstyrning. Available at: https://www.sweco.se/vart-erbjudande/transportinfrastruktur/jarnvag/trafikstyrning/ (Accessed: 16 March 2026).

Toro, J. V., & Tarkian, M. (2025). Optimizing Text Recognition in Mechanical Drawings: A Comprehensive Approach. *Machines, 13(3)*, 254.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2020) Training data-efficient image transformers & distillation through attention. [Online].

Trafikverket. (2020). *Signalställverk modell 59*. [Online]. Available at: https://bransch.trafikverket.se/for-dig-i-branschen/teknik/anlaggningsteknik/signalteknik/signalstallverk-modell-59/ (Accessed: 16 March 2026).

Trafikverket (2023). *ATC – tågskyddssystem*. [Online]. Available at: https://bransch.trafikverket.se/for-dig-i-branschen/teknik/anlaggningsteknik/signalteknik/atc--tagskyddssystem/ (Accessed: 16 March 2026).

Trafikverket (2025). *TRVINFRA-00301 Krav med rådstext: Signalsystem – Projektering allmänt* (Version 17.0, Publication date: 2025-11-01). Available at: https://puben.trafikverket.se/dpub/visa-dokument/45476048-dead-47b2-9e85-04856cb4fed6 (Accessed: 19 February 2026).

Ultralytics (2026). *Ultralytics YOLO models*. [Online]. https://www.ultralytics.com/yolo (Accessed: 18 May 2026).

WWWInsights (2024). How to Measure the Performance of OCR: Why BLEU Isn't Always the Best Choice. Available at: https://www.wwwinsights.com/ai/bleu-ocr/ (Accessed: 19 May 2026).