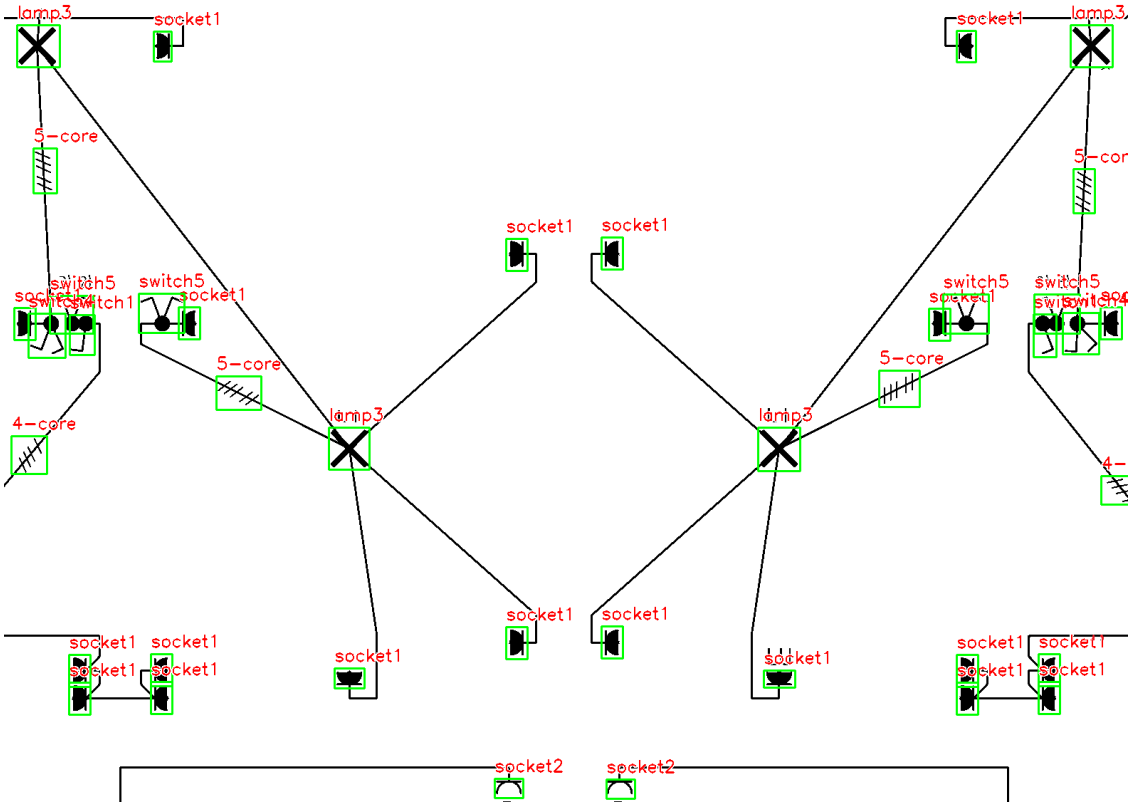


Beyond the Legend

Symbol Localization and Reference-Guided Classification in Industrial Diagrams

Master's thesis in Complex Adaptive Systems

Eddie Ström

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Beyond the Legend

Symbol Localization and Reference-Guided Classification in Industrial Diagrams

Eddie Ström

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2026

Beyond the Legend
Symbol Localization and Reference-Guided Classification in Industrial Diagrams
Eddie Ström

Supervisor: Ola Löseth, Actemium Energy AI
Examiner: Kristian Gustafsson, Department of Physics

Master's Thesis 2026
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Predicted classifications on symbols in a electrical housing diagram.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Beyond the Legend
Symbol Localization and Reference-Guided Classification in Industrial Diagrams
Eddie Ström
Department of Physics
Chalmers University of Technology

Abstract

The digitization of industrial diagrams has seen increasing attention across many engineering domains due to these documents often being the backbone for downstream applications such as maintenance, revision work and validation. But working with these drawings can often times be expensive, time-consuming and repetitive. While related work has explored automated symbol detection in domains such as piping and instrumentation diagrams (P&IDs), electrical housing diagrams are less studied even though they have many of the same challenges. This thesis investigates symbol localization and reference-guided classification in electrical housing diagrams, focusing on the use of diagram legends as references.

The work compares two methodologies, a more traditional template matching approach and a two-stage approach, where symbol regions are detected, by a generic symbol region of interest detector, and then classified using legend-based references. The two-stage method is tested using both template matching and Siamese-network-based classification. By relying on legend symbols, the proposed methods reduces the dependence on annotated training datasets reflecting realistic scenarios where data, more often than not, is very limited. The generic regions of interest detector was able to localize a large portion of the relevant symbols, reaching a maximum recall of 95.42%. Classification on the proposed regions showed promising performance, with results comparable to the traditional template matching approach. The reference-guided two-stage method also offers a more flexible structure by separating localization from classification as well as incorporating deep-learning models, providing stronger potential for scalability, speed, refinement and versatility.

Keywords: Industrial Diagrams, Computer Vision, YOLO, RT-DETR, Faster-RCNN, Object Detection, Siamese Network

Acknowledgements

I would like to thank everyone at the Actemium Energy AI team at VINCI Energies for the constant help and support that I have received throughout the project. Léon Löwered, Jari Kesti, Marie Kurtin and Chanin Lerdmaleewong have all aided me with this project, and a special thanks goes out to Ola Löseth, Tomas Grahn, and Erik Sahlin who has provided extra guidance during my time here. I would also like to thank my examiner Kristian Gustafsson for all the help I received during this endeavour.

Eddie Ström, Gothenburg, May 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CNN	Convolutional Neural Network
COCO	Common Objects in Context
DETR	DEtection TRansformer
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Networks
FRCNN	Faster Region-Based Convolutional Neural Network
GT	Ground Truth
IoU	Intersection over Union
MHSA	Multi-Head Self-Attention
NMS	Non-Maximum Suppression
P&IDs	Piping and Instrumentation Diagrams
RoI	Region of Interest
RPN	Region Proposal Network
RT-DETR	Real-Time DEtection TRansformer
TP	True Positive
WBF	Weighted Box Fusion
YOLO	You Only Look Once

Contents

List of Figures

List of Tables

1
Introduction

The digitizing of engineering schematics has gotten more attention over recent years [5]. In many engineering domains, schematic documents are used for design, maintenance, validation and revision processes. A big problem however is that a lot of the information is still extracted and handled manually. This makes for slow, inefficient processes, especially when large volumes of existing documentation must be updated, checked or transferred into digital systems.

Promising results of digitization steps have been reported for a wide range of different diagram types [5]. One of the types is Piping and Instrumentation Diagrams (P&IDs), which are schematics used in process industries to map the physical and functional connections between piping, instruments and mechanical equipment. Due to the high volume of legacy P&IDs stored in physical or unstructured digital formats, automated extraction methods are frequently researched to convert these documents into machine-readable data for facility management. In this context, machine learning methods has been used to detect and classify symbols in complex technical drawings. Additionally, reference-guided and few-shot approaches have shown that symbol detection can be formulated such that the dependence on large annotated datasets can be minimized. These methods make use of known symbol references and visual similarity, making them well suited to contexts where annotation can be costly or has a very limited availability of data [6].

Related works on electrical diagrams has also explored symbol and component detection through supervised object detection with a study testing You Only Look Once (YOLO) based systems [7]. YOLO operates as a single-stage detector that divides an image into a grid, processing it in a single pass to both predict bounding box coordinates and object classifications [8]. The results show how recent detectors can be adapted to engineering drawing analysis outside the P&ID domain [7]. This is relevant because it shows that electrical diagram interpretation can be seen in a similar way to other technical drawing tasks, but it also shows that there currently is a common dependency on predefined classes and training data that needs to be specific to the target document type.

Like P&IDs, electrical housing diagrams are information-dense technical drawings containing standardized symbols, textual annotations and structural relationships. At the same time, they are often visually cluttered, affected by noise, and made out of repeated elements that can make automatic interpretation difficult. Many of these diagrams include a legend, which specifies what symbols are used in the

drawing as well as associated symbol information. This provides an opportunity to see the problem as a reference-guided symbol detection problem, where symbol examples from the legend are used to find matching instances in the full diagram.

1.1 Motivation

Schematic documents are used in a lot of engineering industries, and many times they are used as the base for downstream tasks. Working with such drawings is often very time-consuming and repetitive, especially when changes must be identified manually or when legacy documents needs to be digitized.

Electrical housing diagrams are usually subjected to this problem as they often have large number of symbols, labels and structural elements that many times are dense, noisy and visually complex. This makes them difficult to process automatically and expensive to update by hand, and as such, there is motivation to develop methods that can support engineers in automating parts of the interpretation process. This makes it possible for people to focus on other aspects, such as analysis, decision-making and quality control and not repetitive document handling.

1.2 Research Questions

This work investigates whether the reference-based ideas that have shown promise in domains such as P&IDs can be extended to electrical housing diagrams, specifically those following the Swedish standard. The objective is to develop a method that can identify corresponding symbol instances in a complete diagram, given a legend containing example symbols and symbol information. Due to the common constraint of having very little annotated data available, all annotated diagrams shall be used only for inference, with the exception for the corresponding diagram legends. The symbols extracted from the legend can be used either as reference templates or as examples that are augmented to synthetically generate training data.

- **RQ1:** How can a symbol detection system identify and classify components in electrical housing diagrams using only isolated reference examples from a diagram's legend?

- **RQ2:** How can synthetic data and augmentation of legend symbols be leveraged for detection when having a very limited dataset?

2
Background

This chapter presents the theoretical background and explanations over methodologies, models, algorithms and concepts used in this thesis. It starts with traditional computer vision detection and then moves on to both convolutional neural networks and transformer-based object detection. It also explains methods on how to work with and fine-tune models, generating synthetic data, and ends with methods for evaluating results.

2.1 Traditional computer vision object detection

A common trait for traditional object detection is using sequences of processing stages, combining multiple techniques over different steps. For technical drawings this can be binarization, edge extraction, connected-component analysis, contour tracing as well as morphological operations such as dilation and erosion [9, 10, 11]. It is also possible to do more shape-oriented preprocessing like skeletonization [12]. The operations themselves do not detect objects, but they can convert a dense and complex diagram to more digestible information and provide candidate regions and features that can be easier to isolate and compare. There are existing methods for extracting local directional changes and topological markers, pool them into a feature vector, and use the representation as input to for example linear classifiers [10, 13].

Advantages of having multi-stage pipelines is transparency and interpretability, where the different steps can be visualised and tuned. This is relevant in many cases, such as when image quality can vary a lot across documents, leading to difference in thresholds [14]. Traditional vision-based pipelines are still used for problem-specific solutions [13, 15].

2.1.1 Template matching

Template matching is a classical pixel-based detection approach where a fixed reference template is compared to different regions in a larger image. One method for querying these regions is using a sliding window, where the template is slid over the search image and a similarity measure such as normalized cross-correlation, is evaluated at each step [16]. The similarity scores create a response map, where local maxima indicate any potential detections. It has been shown that variations of template matching is viable for finding queried symbols in technical plan diagrams

[17].

The normalized correlation coefficient, as implemented in OpenCV [18], an open-source computer vision library, is explained below. Let $T(x', y')$ denote the template of size $w \times h$ and $I(x + x', y + y')$ the image patch under the template when the template is placed at location (x, y). The mean-subtracted template and image patch are defined as follows. Here, (x') and (y') denote coordinates within the template, while (x") and (y") are summation indices over the full $(w \times h)$ template-sized region. For the template term, this region is fixed and corresponds to the entire template. For the image term, the same $(w \times h)$ region is evaluated at the current image location $((x, y))$.

$$T'(x', y') = T(x', y') - \frac{1}{wh} \sum_{x'', y''} T(x'', y''), \tag{2.1}$$

$$I'(x + x', y + y') = I(x + x', y + y') - \frac{1}{wh} \sum_{x'', y''} I(x + x'', y + y''). \tag{2.2}$$

The normalized correlation score at location (x, y) is

$$R(x, y) = \frac{\sum_{x', y'} T'(x', y'), I'(x + x', y + y')}{\sqrt{\sum_{x', y'} T'(x', y')^2}; \sqrt{\sum_{x', y'} I'(x + x', y + y')^2}}. \tag{2.3}$$

This normalization compensates for variations in global brightness and contrast. A value close to 1 indicates a strong positive correlation between the template and the image patch, while values near 0 suggest a lack of similarity. Negative values imply reversed contrast where the intensity patterns are essentially opposite [19, 20].

2.1.2 Chamfer matching

Chamfer matching is a contour-based matching method that uses a distance transform to measure how closely the edges of a template align with the edges in an image [21]. A target image is converted into an edge map and a distance transform is applied such that every pixel is assigned a value representing its distance to the nearest edge. Then a template of contour points slides across this transformed image. At each location the algorithm calculates a match score by summing up the distance values underneath the template's points [22].

Let $Q = \{q_i\}_{i=1}^{N}$ be the set of contour points in the template, $D_I(\mathbf{x})$ is the distance transform of the target image contour map [22]. For a translation $\mathbf{t}$, the chamfer score is calculated by averaging the distance values at the displaced template coordinates [21]. This is written as

$$d_{\text{chamfer}}(Q, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} D_I(q_i + \mathbf{t}). \tag{2.4}$$

Low scores indicates that the template points are close to observed image edges, and high scores mean poor alignment [21]. The search space is often expanded over scale and rotation parameters due to trying to get geometric invariance [22].

2.2 Convolutional neural networks-based object detection

A lot of modern object detection methods rely on deep learning models that are capable of end-to-end training and automated feature extraction. A very popular such model is Convolutional Neural Network (CNN), which learns to extract key features from an image by filtering raw pixels layer by layer, transforming them into meaningful representations [23, 24]. Using convolutional layers, these networks extract spatial and hierarchical characteristics, going all the way from basic edges and textures to more complex semantic shapes. This makes it possible to identify patterns in more dense and noisy environments [25]. The feature representation assumes that neighbouring pixels are related and that patterns can be recognized regardless of their position [26].

Modern CNN-based detectors are usually structured into three main components, a backbone network, a neck and a detection head [26]. The backbone is the core feature extractor, using architectures to generate high-level feature maps from the input data. These features are then processed by the detection head, which does spatial localization, through bounding box regression and object classification by assigning probabilistic class labels to candidate regions. The neck module sits between the backbone and head to refine and combine extracted features [27, 26]. An upside with this architecture is that it has the ability to use transfer learning, where networks pre-trained on large-scale datasets are fine-tuned for specialized tasks, reducing the computational cost and data requirements for domain-specific applications, for example electrical symbol detection [25].

2.2.1 You Only Look Once (YOLO)

You Only Look Once (YOLO) is a highly used method in object detection and it can see spatial localization and semantic classification as a single end-to-end regression problem. YOLO takes an input image, divides it into $S \times S$ grids. For each grid cell, the network predicts B bounding boxes as well as giving confidence scores and class probabilities in a single forward pass. YOLO then uses non-maximum suppression (NMS) to remove redundant overlapping bounding boxes, keeping the highest-confidence detection for each object [8]. This architecture makes it possible for the model to reason globally about the image and its context [8]. By processing the whole image at the same time, the network can learn to connect objects with their surroundings, which can lower false positive detections on background noise. YOLO also uses a multi-part loss function that punishes localization errors, confidence inaccuracies and classification mistakes.

While early YOLO architectures struggled to detect small objects due to heavy downsampling diluting details, a more popular solution for later models became Feature Pyramid Networks (FPN) [26]. Spatial information loss can occur because early network layers preserve pixel-level boundaries but do not contain enough semantic information to identify objects. Deeper layers on the other hand contain stronger

object-level information, but the image representation has often been downsampled to the point where small targets may disappear. The prevention done by FPN is merging the object-aware data from the deep layers with the high-resolution pixel maps from the early layers, which allows the model to place bounding boxes on small objects. This creates a pyramid where all levels have strong semantic meaning, making it possible to detect objects over a wide range of scales and sizes [28]. Additionally, in the newest iterations, steps has been taken to remove the NMS bottleneck for single-box predictions. For YOLO26, additional steps have also been made to improve small object detection with methods like Progressive loss balancing (ProgLoss) and small-target-aware label assignment (STAL) which forces the network to penalize errors on objects with very low pixel counts [29].

Applying YOLO-based detectors to technical drawings has its difficulties that are not necessarily seen in natural image benchmarks such as the Microsoft Common Objects in Context (COCO) dataset [30] or the ImageNet Large Scale Visual Recognition Challenge dataset [31]. Technical drawings are mostly monochromatic and are often times densely cluttered with relatively small symbols. These are in general fundamentally different from photographic imagery. But despite this, together with more modern iterations, YOLO has been found to be useful even within the technical diagrams domain. One study focusing on electrical and engineering schematics has demonstrated that YOLO-based models can successfully detect and classify components in technical drawings, even when trained on limited and synthetically generated datasets [7].

2.2.2 Faster Region-Based Convolutional Neural Network (FRCNN)

The Faster Region-Based Convolutional Neural Network (FRCNN) framework is made for high-precision detection by having a two-stage architectural design [32]. The model uses a Region Proposal Network (RPN) that scans shared convolutional features to identify likely object regions while at the same time estimating how likely each region is to have an object in it. Unlike its predecessors which used computationally expensive algorithms like Selective Search, the RPN instead gives candidate regions from shared convolutional features within an end-to-end trainable detection framework [32]. These proposed regions are then passed to a second-stage detection head, where Region of Interest (RoI) pooling standardizes features from regions of different sizes before final classification and bounding box regression [26].

Research has also successfully used FRCNN-based frameworks for automated detection of symbols in electrical schematics where the model identifies different sized components with connecting lines and text labels [33]. Similar research has seen promising results within P&IDs within the chemical process industry [34].

2.3 Transformer-based object detection

Another method that also has seen a lot of development over many domains is Transformers. This is due to their ability of global context modelling [35]. Transformers use Multi-Head Self-Attention (MHSA), allowing the model to evaluate relationships between image patches or feature tokens over the whole image and not being limited to only local neighbourhoods [36]. This broader view help the model find long-range relationships.

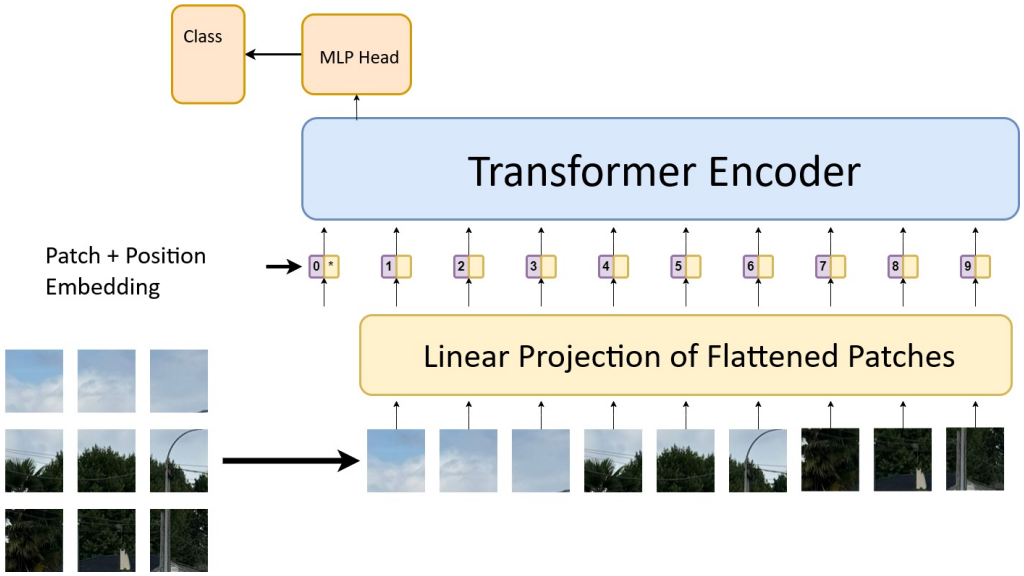


Figure 2.1: The Vision Transformer steps. Image decomposition into patches, linear projection and global attention mapping. Image adapted from [1].

When adapting Transformer architectures to computer vision, the Vision Transformer was introduced to process images using the same general sequence-based structure as language Transformers [37]. As shown in Figure 2.1, the queried image is first divided into fixed-size patches, which are then flattened and linearly projected into embeddings. These are then, combined with positional information and taken through a Transformer encoder. In the encoder layers, MHSA projects each patch embedding into learned query, key and value vectors, which are in turn used to compute attention weights between patches over the image. Each patch representation is updated using information from the whole image, incorporating global contextual modelling [36]. Due to this the model can identify long-range relationships that localized convolutional filters might miss. Learned positional embeddings are added to save spatial information which gives the ability to have global reasoning while still keeping knowledge of the patches' arrangement [37].

In the domain of technical drawings, global reasoning can help systems to identify symbols or structures that look very similar but are of different classes depending on how they are connected or what they are connected to. For example, Transformer frameworks have been successfully applied for segmenting multi-part technical draw-

ings in patent documents [38].

2.3.1 Real-Time Detection Transformer (RT-DETR)

One of the most important features of Transformer-based detectors is that they can treat object detection as an end-to-end global set prediction problem and not using any anchor-based designs [39]. Transformer-based methods has self-attention over the image and uses it to give a fixed set of detections, which makes the model reason about relationships between objects. DEtection TRansformer (DETR) implements this and uses learned query embeddings together with a bipartite matching loss based on the Hungarian algorithm during training. The Hungarian algorithm is used to decide which predicted boxes should be matched with which ground-truth objects in the image. It compares the model's predictions with the annotated objects and finds the best one-to-one matching, so that each object is assigned to at most one prediction [40]. This matching teaches the model to avoid duplicate detections, which means that there is no need to use methods like NMS to remove redundancies. The trained model then, during inference, gives a fixed set of detections, where low-confidence or 'no object' predictions are ignored, simplifying the detection processing [39].

Detection Transformer architecture are limited by the Transformer encoders computational complexity. In Detection Transformers, the computational cost of self-attention scales quadratically with the number of image tokens. These tokens correspond to spatial positions in the image feature map, which are derived from the input pixels. When the input image resolution increases, the number of tokens also increases, causing the attention computation to grow rapidly [41]. This makes traditional DETR slow to train and often needs several more epochs to converge when compared to CNN counterparts, and it is also inefficient at real-time inference [41].

To manage this computational bottleneck the Real-Time DEtection TRansformer (RT-DETR) was created, having good performances and compares to modern YOLO variants in real-time domains [42]. RT-DETR uses a Efficient Hybrid Encoder, which changes feature interaction to minimize the cost of self-attention without sacrificing global context modelling. RT-DETR separates feature processing into two components that handle high-level and low-level representations.

- **Attention-based Intra-scale Feature Interaction (AIFI):** Self-attention is only used on high-level semantic feature maps, where the spatial resolution is lower but with more content. This allows long-range dependencies while still lowering the quadratic computational cost that happens with more dense self-attention [42].

- **CNN-based Cross-scale Feature Fusion (CCFF):** CCFF combines the features from different scales using a lightweight CNN-based module. This makes it so that the model can use both detailed low-level features as well as high-level semantic features, all while keeping the efficiency of convolutional

operations [42]. This is faster than full self-attention that compares each position in the feature map with every other position.

RT-DETR uses an Intersection over Union (IoU) aware Query Selection strategy which prioritises higher-quality object candidates for the decoder, based on localization confidence which is a measure on how accurately the bounding box manages to frame the object. This improves convergence speed and bounding box regression accuracy compared to earlier DETR variations [42].

The RT-DETRv2 is an additional improvement of the RT-DETR architecture that focuses on improved training efficiency and more flexible deployment in more resource constrained environments [43]. The RT-DETRv2 uses a selective multi-scale feature extraction that distributes sampling points more proportionally across feature levels based on differences in resolution and feature scale. Due to it being a discrete sampling operator, it reduces the number of sampled spatial locations, lowering the amount of interpolation required and decreases computational cost [43].

For technical drawings and engineering schematics, DETR models can have advantages over one-stage detectors. Industrial diagrams can at times be cluttered, requiring models to differentiate between small geometric differences based heavily on surrounding context and connectivity [44]. While some YOLO architectures can have problems with lower spatial precision when handling complex and overlapping geometries [45], the global receptive field of the RT-DETRs hybrid encoder can give it the ability to capture context and relationships across the entire document.

2.4 Siamese Architectures

The object detection problem can also be solved learning a distance function between pairs of inputs, making the classifications by identifying whether samples are of the same class or not, based on similarity. A known architecture for this method is the Siamese Neural Network, which consists of two or more identical sub-networks that have the exact same configurations, parameters and weights [46]. A visual representation can be seen in Figure 2.2 below. In a forward pass, a set of input images is fed into the network and each branch maps its respective input to a high-dimensional feature representation $f(x) \in \mathbb{R}^d$. Due to the weights being shared, the network is always going to be symmetric, meaning that two very similar inputs will consistently be projected close together in the embedding space [46]. Classification is then performed by computing the distance between these feature representation.

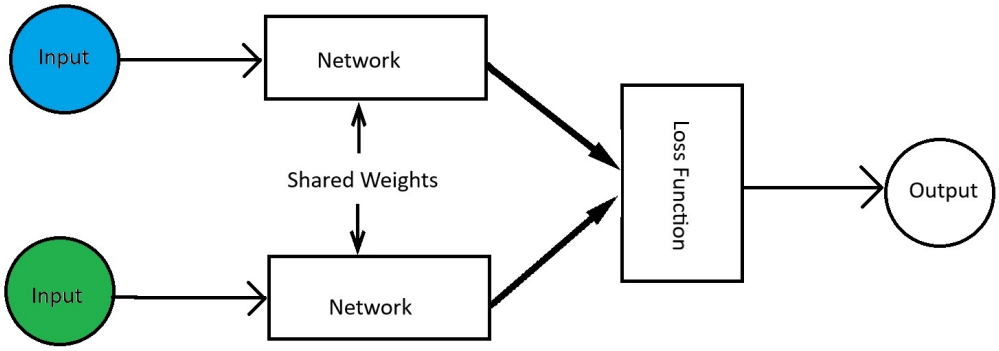


Figure 2.2: Visual representation of a Siamese Neural Network given two queries.

A recent study have used Siamese architectures to match diagram symbols against their corresponding legend present in P&ID schematics. Because diagram-legend entries give typical examples of how the symbol looks like in a specific schematic, models can use these pairs to encode feature representations from the data. Treating the diagram-legend as a classification baseline the network gets a localized embedding space where diagram symbols are grouped around their respective legend anchors [6]. This strategy has also been seen as effective in broader semi-supervised pipelines, for example, another study [47] has shown that having a initial class-agnostic object detector can locate potential symbols throughout a document. Then a Siamese network performs label transfer by computing the similarity between the unlabelled regions of interest and a small set of annotated instances, which generates pseudo-labels for the remainder of the dataset. The ability of these networks to cluster together similar symbols while separating non-similar ones depend on a loss function that makes sure that similar samples are mapped close together and other samples are spread apart during training.

2.4.1 Triplet Loss Function

A common loss function being used for training Siamese Networks is Triplet Loss [6, 47]. This loss function evaluates three samples at the same time to pull matching samples together and push non-matching samples away. The three samples taken are denoted as follows: an Anchor (a), a Positive (p) and a Negative (n) where p is a sample of the same class as the anchor, and n is a sample of a different class [48]. The objective of the loss function is to pull the anchor and positive embeddings together while pushing the negative embedding away by a predefined margin. This is to have the distance between the anchor and the positive smaller than the distance between the anchor and the negative. The loss function L for a triplet is defined as following

$$L = \sum_{i}^{N} \max\left(\|f(a_i) - f(p_i)\|_2^2 - \|f(a_i) - f(n_i)\|_2^2 + \alpha, 0\right)$$

where $f(\cdot)$ is the embedding function and α is the predefined margin, which helps the network to not converge at a trivial solution where all embeddings are identical [48]. The effect of Triplet Loss on the embedding space is shown below in Figure 2.3.

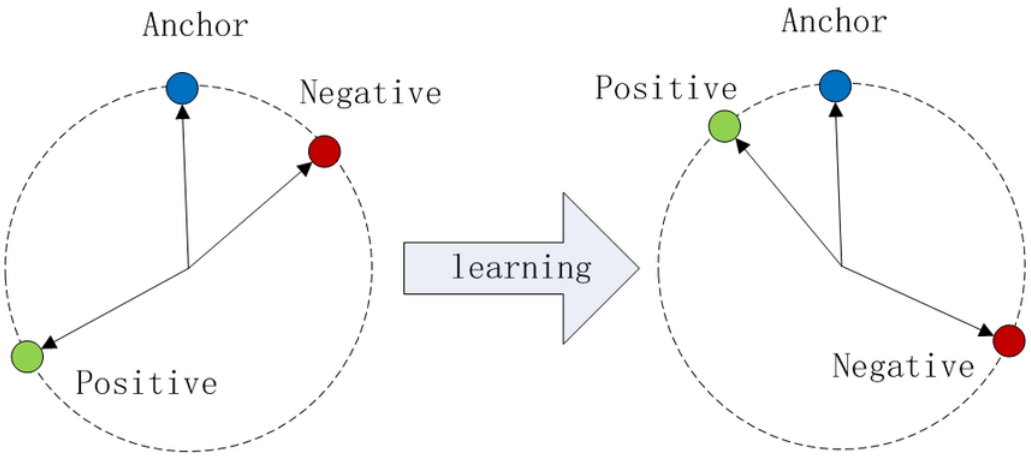


Figure 2.3: Before training, anchor, positive and negative samples can be separated in a non-optimal way in embedding space (left). After training, embeddings of the anchor and positive are pulled closer together while negatives are pushed (right). Reproduced from [2]. Licensed under CC BY-SA 4.0.

One of the issues with Siamese networks is that during training, if the input triplets are sampled completely randomly, many of them will be within the distance constraint ($d(a,p)+\alpha < d(a,n)$). This gives a loss value of zero and less contribution to parameter updates [48]. Studies have shown that training can benefit from prioritizing samples that cause non-zero loss, which is why strategies such as hard example mining exists [49]. In triplet loss, one can use Hard Negative Mining, where the triplets are selected based on their relative difficulty in the current embedding space [48]. In the same study, two types of informative negatives are distinguished

- **Hard Negatives:** Triplets where the negative is actually closer to the anchor than the positive is ($d(a,n) < d(a,p)$). These give high loss and can force the model to make large changes.

- **Semi-Hard Negatives:** Triplets where the negative is further from the anchor than the positive, but still within the margin α ($d(a,p) < d(a,n) < d(a,p) + \alpha$).

2.5 Methods for Fine-tuning large models

Large, pre-trained neural networks needs fine-tuning to work effectively on more specific, specialized tasks or domains [50]. Training a deep learning architecture from scratch tend to need a vast amount of data as well as computational resources, so instead one can use transfer learning. The model is then initialized with weights

already learned from a larger dataset for a different purpose, before being trained on another dataset from the targeted domain [50]. Models such as Faster R-CNN, RT-DETR and YOLO are usually already pre-trained on large-scale benchmark datasets such as Microsoft COCO [30] which contains over 330,000 images and 80 object categories [51].

2.5.1 Layer Freezing

In a convolutional or transformer-based network, the earlier layers typically extract more generic visual features, while the deeper layers learn more specific patterns. By using layer freezing technique, it is possible to freeze the weights of the early layers. The gradients are then not calculated and weights are not updated during backpropagation. This leads to the general features being saved and lowering the computational costs [52].

More recent studies on real-time object detection architectures, such as YOLO, show that freezing the backbone can save general-purpose features and reduce GPU memory consumption by up to 28% compared to full fine-tuning, and often getting equal or better mean average precision (mAP) results [53]. The amount of layers one should freeze is dependent on the targeted dataset. When dealing with limited data or severe class imbalance, one needs to freeze more layers, and with a larger dataset freezing only a few can help the network to learn more task-specific details [53].

2.6 Synthetic Data

The performance of deep learning models in object detection often relies on large, diverse and annotated datasets. To gather and then annotate real-world data is generally a very time-consuming, labour-intensive and expensive process [54]. To help with this, synthetic data has been used as a practical and cost-effective addition for model training [55]. By utilizing simulations and rendering techniques, it is possible to generate large volumes of training images with annotations. The synthetic datasets helps gather controlled variation of data, and for tasks such as object detection, it can help with augmentation of environmental factors, object poses and lighting conditions, which creates diverse scenarios that might be uncommon in real-world data [56].

In domains such as technical diagrams, where getting large labelled datasets of specialized symbols is very difficult, synthetic data can be very useful. Recent research has showed that using synthetic data within technical documents like P&IDs, does not need to perfectly replicate exact target symbols. Instead, by generating data that captures the general gist and visual structure of the drawings, a generalized model can be trained to successfully detect and extract a wide variety of symbols [6].

2.6.1 Methods for Generating Synthetic Data

There are many different methods of generating synthetic data with a efficient method being simple 2D overlay. Here images of target objects are placed onto random or domain-specific background images. This makes it possible for fast generation of large datasets and can sometimes be expanded with 3D rendering or dedicated simulations to better capture complex poses and spatial relationships [56].

For models to generalize well to real-world examples and to not overfit to synthetic generation patterns, synthetic datasets are typically subjected to a lot of data augmentation and randomization. Augmentation techniques include rotation, translation and scale changes, as well as morphological operations like dilation and pruning. Additionally, including visual noise and background clutter to help the network to learn more invariant features rather than relying on clean synthetic backgrounds is also useful [56].

2.7 Performance Metrics

The evaluation of symbol detection measures how well the model localizes symbols with bounding boxes and assigns the correct class labels to the detected symbols.

2.7.1 Intersection over Union (IoU)

IoU is a standard metric for evaluating the overlap between a predicted bounding box (B_p) and the ground truth (GT) box (B_g). It is defined as the area of intersection divided by the area of the union.

$$IoU = \frac{Area(B_p \cap B_g)}{Area(B_p \cup B_g)} \tag{2.5}$$

Within object detection a prediction is typically considered a True Positive (TP) if it has the correct class label and its IoU with the matched GT box is greater than a predefined threshold. Predictions that do not match any GT are counted as False Positives (FP), and GT objects that are not detected counts as False Negatives (FN). A visual representation of IoU is shown in Figure 2.4 below.

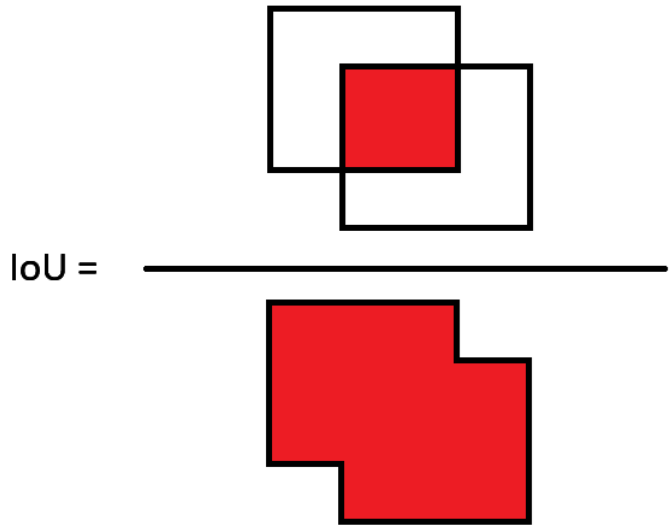


Figure 2.4: Visual representation of IoU for bounding box evaluation.

Predicted boxes need to match GT boxes and is usually done one-to-one, such that each GT object can only be assigned to one prediction. A common way to perform this assignment is to use the Hungarian algorithm, which finds an optimal matching between predictions and GT objects based on a chosen cost, such as IoU. If a prediction is set to the wrong object or does not reach the IoU threshold, it counts as a FP, while at the same time the now unmatched GT object is counted as a FN.

2.7.2 Weighted Box Fusion (WBF)

Weighted Box Fusion (WBF) is a algorithm for combining bounding box predictions for object detection models [57]. WBF uses the information from all predicted bounding boxes to construct a single, more accurate fused box. When multiple boxes overlap an object, the coordinates of the final fused box is calculated as the weighted average of the predicted boxes. For a given coordinate X (e.g., X_{min}), the fused coordinate is computed using the confidence scores C_i and coordinates X_i of the N overlapping boxes.

$$X_{fused} = \frac{\sum_{i=1}^{N} C_i \cdot X_i}{\sum_{i=1}^{N} C_i} \tag{2.6}$$

By using the confidence scores as weights, predictions with high confidence scores influence the final box placement more while less certain predictions still contribute [57]. A visual representation of how multiple overlapping boxes are fused is shown in Figure 2.5 below.

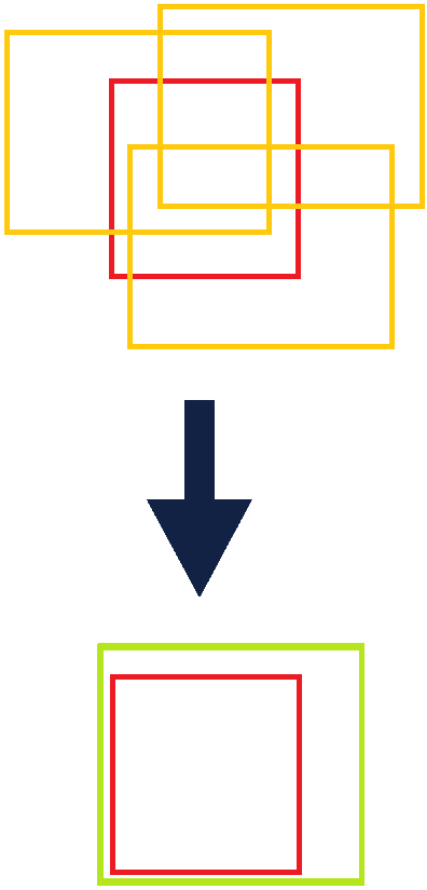


Figure 2.5: Visual representation of Weighted Box Fusion, showing how multiple overlapping predicted boxes are averaged into a single fused box.

2.7.3 Precision, Recall, and F1-score

Precision and recall describe different aspects of detection performance. Precision measures how many of the predicted detections are actually correct.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.7}$$

Recall measures how many of the GT objects are successfully detected.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.8}$$

A False Positive could be a predicted symbol that is not actually present, while a False Negative could be a real symbol that the model fails to detect. Since precision and recall often trade off against each other, the F1-score can be used to summarize them in a single value.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.9}$$

3
Methods

This chapter describes the data curation process, modelling approaches and procedures used in this thesis.

Two main methods are explained. One is a template matching method that performs both symbol localization and classification on whole diagrams. The other is a two step method that first finds general symbol regions of interest using a generic symbol detector that creates bounding boxes around potential symbols. The second step is symbol classification given the aforementioned region of interest bounding boxes.

Classification is performed with respect to the legend of the individual diagram being processed. That is, the set of symbol classes is not defined globally across all legends in the dataset, but is restricted to the symbol types present in the current diagram's legend.

3.1 Dataset description & curation

The dataset used in this thesis consists of 49 annotated electrical housing diagrams, each accompanied by a corresponding legend, either in the same image or in a separate reference document only for the legend. These diagrams were collected from 18 different companies over 34 different projects. The ground truth annotations are in the form of bounding box coordinates and class labels for all identified symbols within a diagram.

All diagrams in the dataset are standardized to a high-resolution format of 8192×5787 pixels. There is a lot of variance between the diagrams in the dataset, as there is differences in age as well as representatives from diverse types of building planning projects, and also rasterization differences. These differences can be quite large between diagrams but can also exist between symbols within the same diagram. Additionally, the style of the drawings and symbols can vary depending on the project and company.

The dataset contains a total of 8,039 annotated symbols distributed across 54 unique classes, but not all annotated symbols in the diagrams are actually defined in their respective legends. Because the legend serves as a reference, a filtering step was used to differentiate what symbols are actually relevant. The refined dataset con-

tains 3,754 total symbols across 37 unique classes. A detailed distributions of the symbol counts before and after the filtering process are shown in Table 3.1. As seen, there is a large dispersion of symbols with high inequality in representation, even going so far as having instances where rare symbols only being shown once over 49 diagrams.

Table 3.1: Statistical summary of symbol counts before and after legend-based filtering.

Metric	**Pre-filtering**	**Post-filtering**
Total number of symbols	7,799	3,754
Number of unique classes	52	37
Per-Diagram Symbol Counts		
Minimum	3.00	3.00
Maximum	506.00	260.00
Mean	159.16	76.61
Median	132.00	62.00
Per-Class Symbol Counts		
Minimum	1.00	1.00
Maximum	2,037.00	1,796.00
Mean	149.98	101.46
Median	60.00	19.00

Shown in Figure 3.1 is an example of an older electrical housing diagram. Figure 3.2 shows a simpler and more modern example.

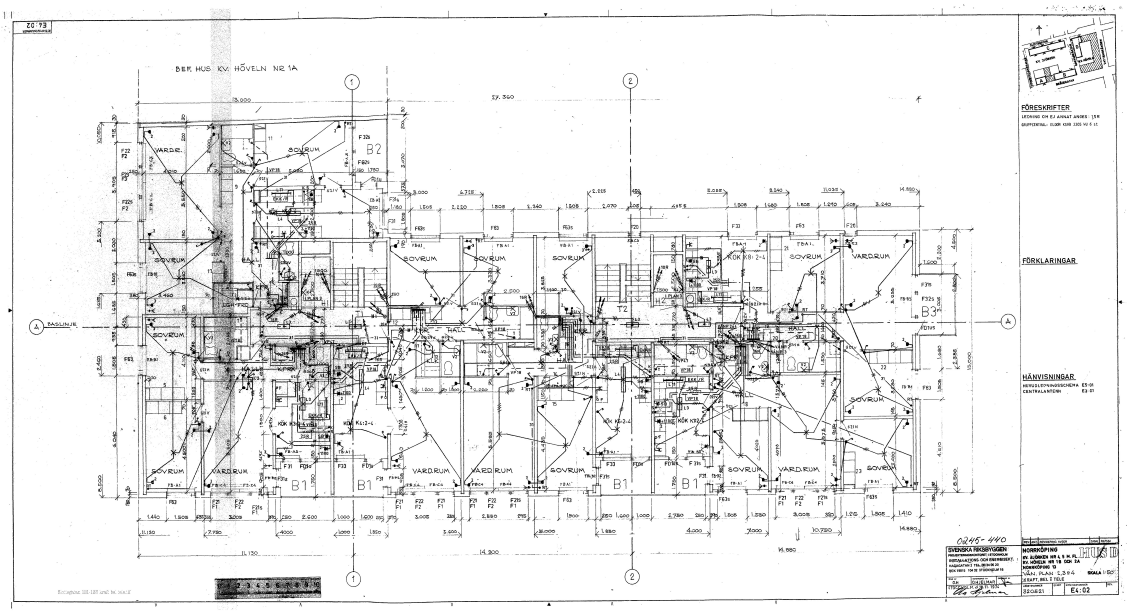


Figure 3.1: Example of an older electrical housing diagram. Here the diagram refers to a separate legend. Reprinted with permission from [3].

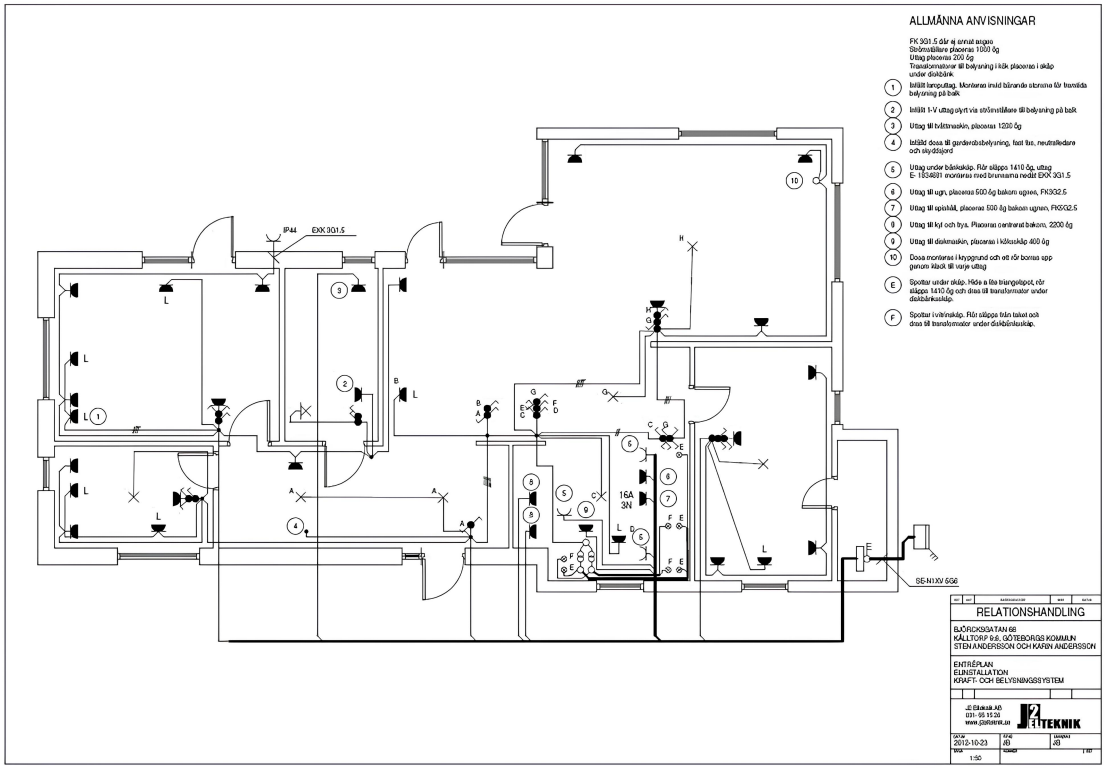


Figure 3.2: Example of a simpler and modern electrical housing diagram. Here the legend is to the right in the image. Reprinted with permission from [4].

3.1.1 Preprocessing

The first preprocessing step made was annotating where the legend is located for every diagram. This was done by placing bounding boxes over these large areas and annotate their coordinates.

Much of the preprocessing relies on a assumption regarding the drawing conventions used. Electrical infrastructure such as wires and symbols are drawn using high-intensity, dark pixels, and background elements such as walls, tables, toilets and other fixtures are rendered using lighter, grey lines. As can be seen in Figure 3.2, modern electrical housing diagrams generally follow this convention. By isolating the darkest pixels, the grey background noise can be filtered out, leaving a clean representation of the electrical system. The filtering process is made using OpenCV, but it does still have limitations, especially when working with older documents. As shown in Figure 3.1 older diagrams can use the same colour and intensity for all lines, which renders this intensity-based filtering ineffective, as no differentiation can be done based on pixel value. The final preprocessing step is text removal. To identify the locations of the text, the images are processed through the Azure Document Intelligence API using Optical Character Recognition (OCR).

An example of an image that has been preprocessed can be seen below in Figure 3.3, where the input image was the previously seen Figure 3.2.

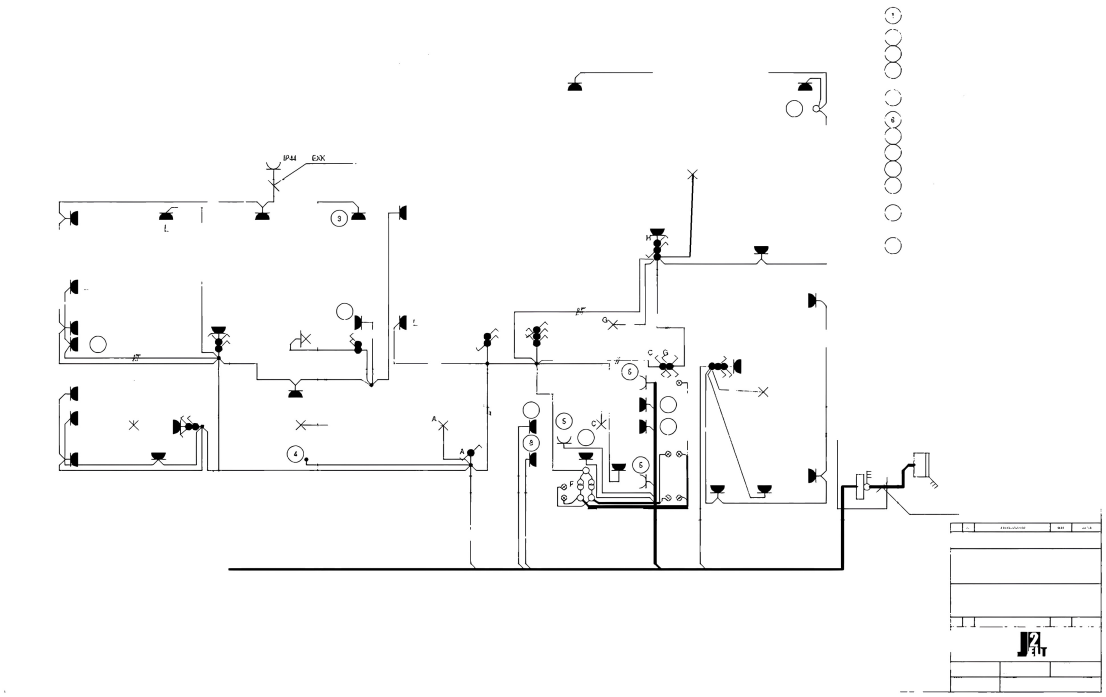


Figure 3.3: Preprocessing on Figure 3.2. Note that the resolution for this image was much lower than what is used in the dataset, and thus has inconsistencies and gaps in the wiring. Adapted with permission from [4].

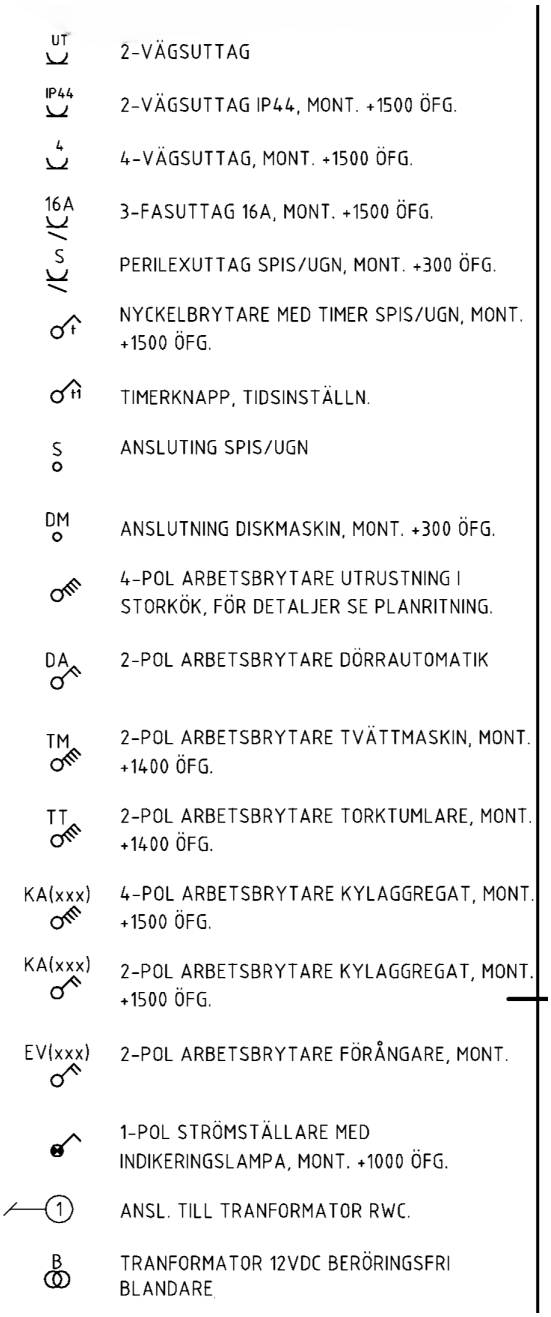
3.1.2 Legend symbol extraction

For downstream training and inference, the legend symbols are extracted, and for this another assumption is made. It is assumed that the symbols are placed on top of each other in rows with their corresponding descriptive text to the right of each symbol. This assumption is held throughout the dataset.

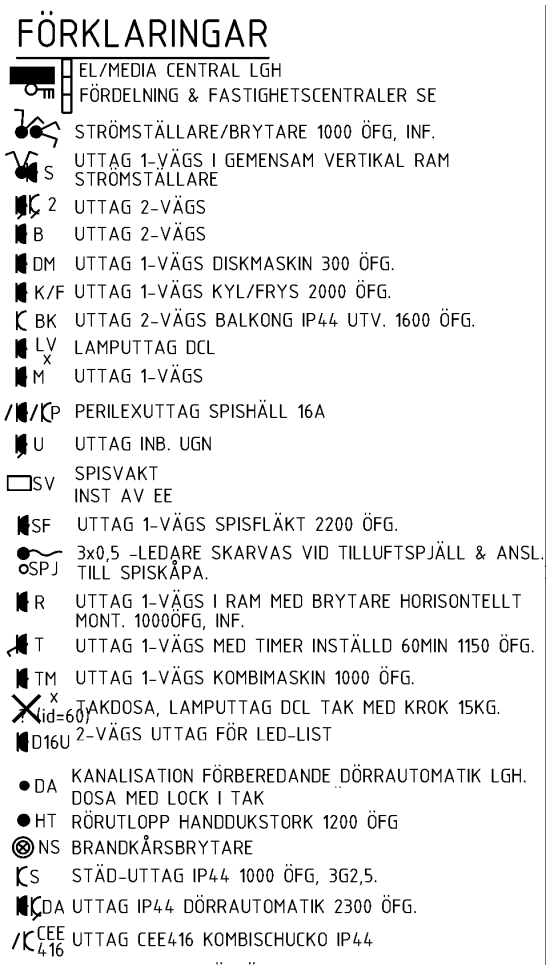
The extraction begins with spatial clustering and topological analysis, using connected components to evaluate geometric complexity, such as topological holes or complex polygonal vertices. Simple, fragmented shapes (e.g., lines or small dots) and larger more complex shapes are localised and differentiated. If a simple shape is within a proximity to a complex shape, their bounding boxes are merged such that multi-part symbols are captured within a single bounding box, even if the different parts are not touching. These bounding boxes were then tightly cropped and stored.

Two examples of a legend containing symbols commonly used in these diagrams are shown in Figure 3.4. As seen in both examples, many symbols are actually identical, with the difference being one or a few associated letters and/or numbers. These are removed during the preprocessing steps and identical symbols then count as the same class. In Figure 3.4a there are 19 symbols, 9 of which are unique, in Figure 3.4b there are 31 symbols, with 15 being unique. To differentiate only the unique

symbols, chamfer matching was used to compare the cropped images with each other. If they passed a similarity threshold then they were considered a duplicate and filtered out. With the same method, the similarity scores of all symbol pairs was stored to pair up very similar but unique symbols for hard negative mining described in later Siamese network training steps.



(a) Examples of symbols in a legend, showing 19 symbols 9 of which are unique.



(b) Examples of symbols in a legend, showing 31 symbols 15 of which are unique

Figure 3.4: Typical legend in electrical housing diagrams.

3.2 Synthetic Data Generation

The synthetic data generation was divided into two categories. The first category was to simulate symbol crops taken from a diagram since the Siamese architecture relies on pairwise similarity and uses the extracted images as anchors for classification. The second category was catered to the larger object detection models evaluated in this study. Because these models need dense spatial context to learn region proposals and bounding box regressions, the generated data was of full-scale diagram simulations.

3.2.1 Synthetic data for Siamese network

The generated synthetic data are of isolated, fixed-size image patches that simulate how symbols may look like if taken from a diagram. Symbol dimensions can vary a lot, ranging from small 20×15 pixel geometries to larger 120×100 pixel bounding boxes. For this, a dynamic rescaling algorithm placed symbols on a canvas such that the largest dimension took up between 60% and 95% of a fixed size canvas to make it more scale invariant due to large possible variations of input queries made from generic symbol region of interest detection made from other models.

Structural noise are placed into the canvases based on properties of real diagrams. Symbols are not completely isolated, they are always physically connected to a wire. So in the synthetic data, a connecting wire was always placed so that it touches the symbol. Included in the images are wire junctions such as 90° turns, T-junctions and multi-directional branching. To simulate nearby component labels and boundaries, the algorithm introduced random text strings and thick, straight boundary lines along the extreme edges of the canvas. Standard data augmentations were also applied. Each symbol had rotation and placement augmentations, morphological erosion jitter was applied to alter the stroke thickness of the lines. A total of 1500 images per class per diagram were generated. Additionally, a separate dataset of pure background noise was also made, simulating clutter and complex wire intersections. To these, procedural box-like geometries, such as rectangles with one wall missing, or four unconnected corners were also added, together with crosses, dots and stray text annotations. Example of an augmented symbol as well as a pure background noise image can be seen in Figure 3.5 below.

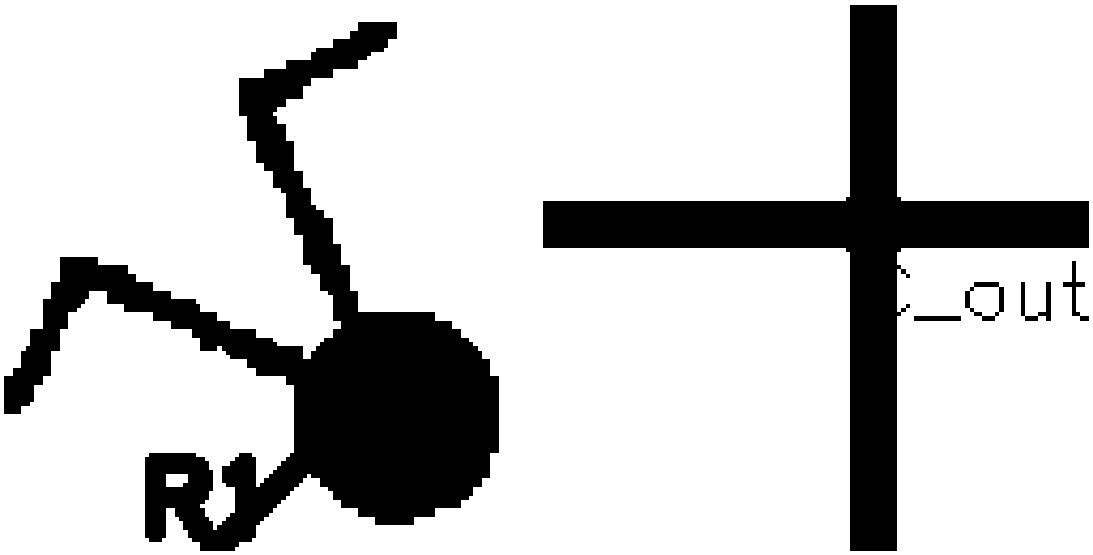


Figure 3.5: To the left an example image of a augmented symbol, and to the right is an example of pure noise, both used in the training of Siamese networks.

3.2.2 Synthetic data for YOLO, FRCNN & RT-DETR

The fully supervised object detection models, YOLO, RT-DETR, and Faster R-CNN need full-scale spatial environments to learn region proposals and context. The generation was made with having these models as generic symbol detectors in mind. All synthetic instances were mapped to a universal "symbol" class.

10,000 of 1024×1024 pixel images were created. Symbols were created from basic geometric shapes, including filled and hollow circles, rectangles, triangles, crosses and lines, examples of which can be seen in Figure 3.6. With the basic shapes, composite structures were made by randomly selecting two from the list, scaling one down and fuse them together. Background clutter was also generated with several intersecting horizontal, vertical and diagonal wires drawn across the entire canvas. Gaussian blur was applied to the entire generated diagram to simulate rasterization. An example of a fully synthesized diagram from this pipeline can be seen in Figure 3.7.

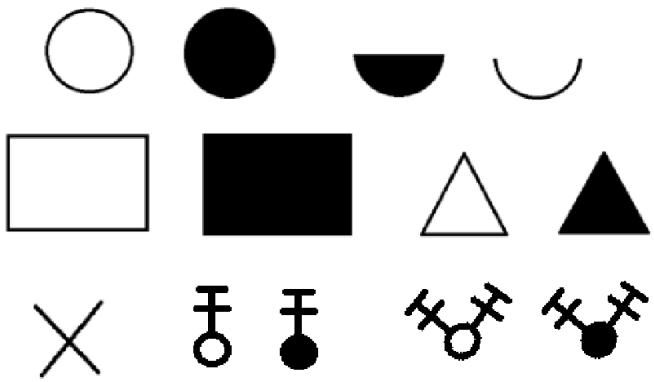


Figure 3.6: Examples of generic symbols used in synthetic data.

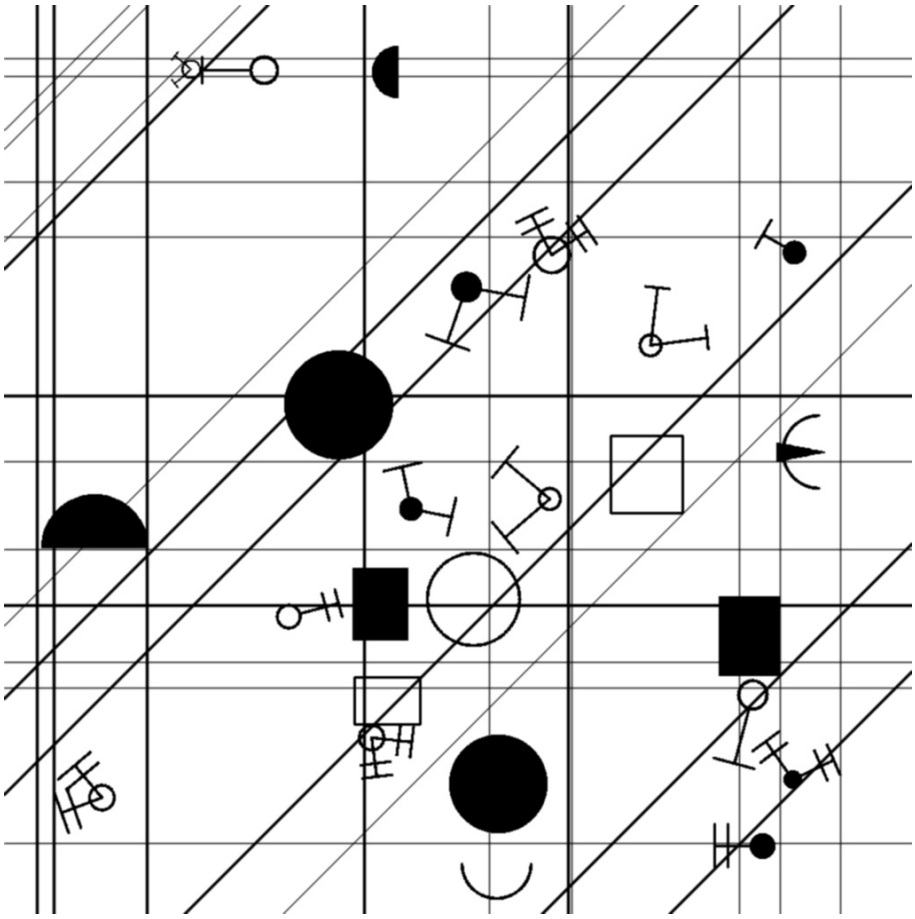


Figure 3.7: An example of a synthesized full-scale electrical diagram. The image features procedurally generated, generic geometric symbols.

3.3 Model training and inference

Training was done on four different architectures, three of which were used for localization and one for classification. YOLO, RT-DETR and FRCNN were all trained and compared on their ability to perform as general symbol region of interest detectors. This was done by seeing the task as single-class detection, having the models isolate symbols from the surrounding diagram background. Different configurations of these were tried during inference for optimal results. As the general region of interest detection is the first step in a two stage process, focus was put on having high recall as to miss as few symbols as possible while keeping relatively tight bounding boxes. This led to having a high number of FPs being of less importance as they should be filtered out during the later classification step. A Siamese Network was then trained for symbol classification given a list of reference symbols. Also, in the two step pipeline, template matching was used in the same way as the Siamese Network for comparison.

3.3.1 YOLO

Sourced from Ultralytics and pre-trained on the COCO dataset, the YOLO architectures were fine-tuned using synthetically generated, full-scale diagrams to function as generic symbol detectors. Training was done on YOLOv8 Small, YOLO11 Medium, and the newer NMS free YOLO26 Medium, for comparison.

The AdamW optimizer was consistently used, having a initial learning rate maintained between 0.0008 and 0.001. The networks were trained for 150 epochs except for YOLO26m, which was extended to 300 epochs. Partial layer freezing was also tested on the YOLOv8s and YOLOv11m models. By freezing the first 10 to 11 layers of the feature-extracting backbone, the models kept foundational visual representations, letting the deeper unfrozen layers and detection heads to specialize on the visual characteristics of the synthetic diagrams. Below in Table 3.2 the different models and hyperparameters tested are shown.

Table 3.2: Summary of YOLO training iterations and hyperparameter configurations.

Model	Frozen Layers	Batch Size	Initial LR	Epochs
YOLOv8s	10	32	0.0008	150
YOLO11m	11	16	0.0008	150
YOLO11m	0	16	0.0010	150
YOLO26m	0	8	0.0010	300

Due to the high resolution of the diagrams they were divided into smaller, 1024x1024 pixel patches using a sliding window with a 20% overlap between adjacent patches. Each patch therefore matched the input image size used during training. The YOLO models generated predictions for each localized patch independently, using a confidence threshold of 0.4. A global NMS with threshold 0.25 was applied to remove overlapping predictions from adjacent patches.

While a initial pass can localise many of the symbols, occasionally multiple distinct but closely spaced symbols can get merged into a single, excessively large bounding box. To mitigate this a targeted, secondary pass was done on the regions predicted by the initial pass. This second pass made on these patches were at a much stricter confidence threshold of 0.85. If the model detected multiple valid sub-components within the patch that met relative area constraints, the original, oversized bounding box was discarded and replaced by the newly discovered boxes. A final NMS pass with an IoU threshold of 0.55 was applied to the combined set of original and refined boxes.

3.3.2 FRCNN

The FRCNN was implemented using the torchvision library and initialized with weights pre-trained on the COCO dataset. Since the objective was class-agnostic symbol localization, the standard multi-class head was replaced with a Fast R-CNN

predictor configured for binary classification. The model therefore differentiated only between foreground symbols and background regions.
The model was optimized using Stochastic Gradient Descent. It was trained for a maximum of 150 epochs, using a `StepLR` scheduler that decayed the learning rate by a factor of 0.1 every 50 epochs.

Similar to the YOLO, the high-resolution, full-scale schematic diagrams were split using a sliding-window to feed smaller sized image patches into the network. Several tests were made, a single-pass inference using a slice size of 1024x1024 pixels with a 20% overlap. Another test was with a smaller slice size of 640x640 pixels with a 30% overlap. A multi-pass, multi-scale method was also tested. The diagrams were processed independently at both the 1024 and 640 scales and all resulting bounding boxes were mapped back to the global coordinate system of the original image. Initially, these overlapping predictions were consolidated using standard NMS. Because multi-scale inference inherently generates a high number of overlapping, slightly offset bounding boxes for the exact same symbol, standard NMS which removes all except for the single highest-confidence box was sub-optimal looking only at recall. WBF was also tested as a replacement to NMS as it utilizes all overlapping boxes to compute a final, averaged bounding box. The coordinates of the fused box are weighted by the confidence scores of the boxes.

3.3.3 RT-DETR

The Transformer-based RT-DETRv2 model, was trained using the Hugging Face ecosystem. The network was initialized using the pre-trained `PekingU/rtdetr2r_50vd` checkpoint, which uses a ResNet-50 architecture as its feature-extracting backbone. The model was fine-tuned for 20 epochs, and to maintain stability and prevent the pre-trained backbone weights from being catastrophically distorted during the initial phases of fine-tuning, a learning rate warm-up for the first two epochs was used.

The inference pipeline for the RT-DETR model uses a similar sliced-inference method as the YOLO architecture. Because transformer-based architectures can be sensitive to spatial transformations and require more intensive computational overhead, the pipeline includes Test-Time Augmentation. During the initial pass, the full-scale diagrams were divided into 1024x1024 pixel slices, and to maximize detection account for varying symbol orientations, a 50% overlap ratio was used alongside Test-Time Augmentation. Each slice was evaluated across five geometric variants including horizontal and vertical flips as well as rotations. The bounding box predictions from all augmented variants were mapped to global image coordinates, and using NMS with IoU threshold of 0.60.

Also here, a second pass was made on the bounding boxes that were made from the first pass exceeding the size limit of being larger than the 25th percentile. Unlike previous iterations of the second pass, here the isolated regions were expanded with a large contextual pad and then placed centrally onto a large white canvas. Because the symbols on these standardized canvases appeared relatively small against the

white background, the confidence threshold for the refinement pass was lowered to 0.10.

3.3.4 Siamese network

An independent Siamese network was trained for each individual diagram and exclusively on the symbol classes defined in that specific diagram's legend. The network used a ResNet-18 backbone for feature extraction, the fully connected classification layer of the ResNet was removed and replaced with a specialized projection head that maps the extracted spatial features into an $L2$-normalized, 128-dimensional embedding space. The network's weights were updated using the Adam optimizer, together with a learning rate scheduler that reduced the learning rate upon validation plateaus, and using a triplet margin loss function.

Two different strategies were used during training for comparison. A two-tiered negative mining strategy and a pure semi-hard negative mining. For the two tiered approach, the first step was at the data-loading level, a custom batch sampler was used to pit known hard-negative classes that share high visual similarity into the same batch. These pairs were created using chamfer matching during the legend symbol extraction step. Afterwards a semi-hard negative mining algorithm was applied during the forward pass. For the second approach, the custom batch sampler was omitted, only using semi-hard negative mining, but still using a large enough batch size (256) at each training step to guarantee several examples from each class within a batch.

3.4 Template matching for Localization and Classification

A classical computer vision approach was implemented using template matching. For this method, an additional assumption had to be made, the scale of the diagram symbols is assumed to be roughly the same as the templates created from the legend. However, the rotation is going to be different, symbols will appear in at least 90-degree increment rotations, with some having fully continuous 360° orientation.

The process begins by generating a template bank for each individual diagram. Each base template is scaled to three sizes (95%, 100% and 110%) and rotated continuously from 0° to 355° in 5° increments.

When the template bank is generated, a sliding-window detection goes through the preprocessed diagram. The similarity calculation uses Normalized Cross-Correlation and a minimum correlation threshold of 0.80 is used for all matches. Because a single template sliding over its exact match will generate several overlapping bounding boxes with high correlation scores a local NMS is used. The NMS uses an IoU threshold of 0.30 to remove redundant adjacent boxes generated by the same template variation. A size-based heuristic is added to the correlation scores giving a

slight advantage in score to larger template matches for instances where a small symbol might incorrectly be detected inside of a larger symbol that is a composite of sub-symbols. A final global NMS pass is made to fix overlaps, only keeping the highest-scoring bounding box for any given physical region also set as IoU 0.3.

3.4.1 Template Matching for Classification

As an alternative classification step, template matching was also used in the same way as the Siamese Network. Three different iterations were tested, each with some changes, trying to maximize performance.

Just like in the previous template matching using sliding-window step, for each symbol class, the available legend crops were expanded into several template variations by applying small scale changes and rotations. Each predicted region of interest was compared against all templates belonging to the same diagram using normalized cross-correlation. The best scoring template was selected as the predicted class, provided that the matching score was over a predefined threshold. NMS was used after matching to remove duplicate predictions and the final detections were saved with their bounding box coordinates, confidence score as well as assigned class label. This iteration was denoted 'Native' template matching due to the scoring relying on the OpenCV template matching library.

A second template matching variant was also tested, where larger templates were given a small preference during the matching step. The original correlation score was still required to pass the same threshold, but templates were compared using an adjusted score that added a size-based bonus that was in proportion to how much of the region of interest was covered by the template. This was done due to many symbols being composites of sub-symbols, where the subsymbol would otherwise receive a higher score than the larger composite.

The final template matching variant replaced the greyscale comparison with a multi-channel representation of both the templates and the predicted regions. Each image was converted into three complementary channels before matching, one channel looked at proximity to edges, one represented the symbol foreground more after small morphological corrections, and the last highlighted corner-like structures. Template matching was then performed on this combined representation rather than on the original greyscale image.

3.5 Evalutation

The YOLO, RT-DETR and FRCNN models were evaluated on their performance to place bounding boxes on regions of interest. Since these models are used as the first step in the pipeline, the main objective was to detect as many true symbol regions as possible. Here the bounding box quality was not deemed to be of the highest importance since even a somewhat suboptimal bounding box would still include the

target symbol and could be dealt with during later classification steps.

For each diagram, the predicted bounding boxes were compared with the ground truth annotations. Predictions and ground truths were paired using the Hungarian algorithm, where valid matches were selected based on IoU. To simulate the fact that even non-perfect bounding boxes should be allowed, dynamic threshold was made. Symbols with small ground truth bounding boxes allowed for much larger predicted regions, with this allowance diminishing as the size of the ground truth increased. From the predictions, precision, recall and F1-score were calculated for each image and then accumulated over the full dataset.

The classification models were evaluated using a very similar but stricter version. A prediction here had to both overlap a ground truth symbol and provide the correct class label. Predicted boxes were first paired up with ground truth boxes using the Hungarian algorithm with a match only counting if the IoU was above a threshold of 0.20.

After spatial matching, the predicted class was compared with the ground truth class. A matched prediction with the correct class counted as a TP, if incorrect the prediction was treated as a FP for the predicted class and at the same time as a FN for the true class. Ground truth symbols without any matched prediction were counted as FN, while predictions without any matched ground truth were counted as FP.

Because the sliding window template matching approach performs both detection and classification in a single pass, it was evaluated using the exact same strict criteria as the classification models.

4
Results

In this Chapter the results are presented. Two different methodologies were tested in this thesis, with one of them being a two step method, step one localization and step two classification. As such, the results are sectioned into three parts, 4.1 showcasing the results for localization and classification using sliding window template matching. Section 4.2 presents the localization prediction results for the two step method, followed by section 4.3 on classification based on the localization proposals.

4.1 Localization and Classification with Template Matching

Template matching is introduced and explained theoretically in Section 2.1.1. Table 4.1 presents the localization and classification metrics for the sliding-window template matching approach. The 37 distinct symbol classes are grouped into 11 broader categories based on the type of component they represent and their structural similarity.

Table 4.1: Object detection performance of the sliding window template matching approach. Results from the 37 symbol classes have been grouped together.

Category	TP	FP	FN	Precision	Recall	F1-Score
Button	17	9	2	0.654	0.895	0.756
Connector	207	444	151	0.318	0.578	0.410
Detector	45	49	32	0.479	0.584	0.526
Impuls	20	6	35	0.769	0.364	0.494
Lamp	230	154	110	0.599	0.676	0.635
Power Station	40	76	11	0.345	0.784	0.480
Socket	1921	671	362	0.741	0.841	0.788
Switch	127	682	259	0.157	0.329	0.213
Transducer	8	8	9	0.500	0.471	0.485
Transform	3	6	1	0.333	0.750	0.462
Misc	18	1658	146	0.011	0.110	0.020
Overall	**2636**	**3763**	**1118**	**0.412**	**0.702**	**0.519**

As can be seen above, there is a large class imbalance, but the method demonstrates

strong localization and classification capabilities for some categories. This is shown by the high recall (0.841) and F1-score (0.788) in the Socket category as well as promising results for the Button category. The socket category is skewed due to the 'socket1' class having 1796 instances over the dataset, magnitudes more than any other symbol, three examples of the symbol is seen in Figure 4.1. Some symbols have very poor results, notably Misc, and Switch. The Misc class being term for square or rectangular symbols often containing internal text and looks very similar to noise, as can be seen in Figure 4.2 where the left image shows a class representative and the right an example of when wires got misclassified. Also showcased in Figure 4.3 are examples of both the Socket and Switch class, with the top two rows being sockets, and the rest switches.

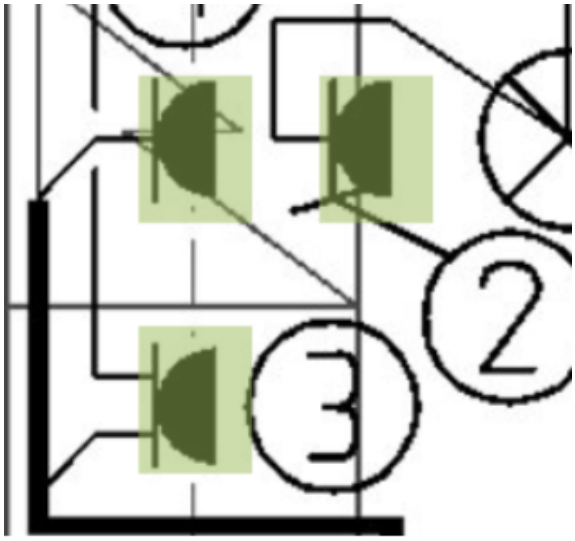


Figure 4.1: Examples of the 'socket1' class, highlighted in green.

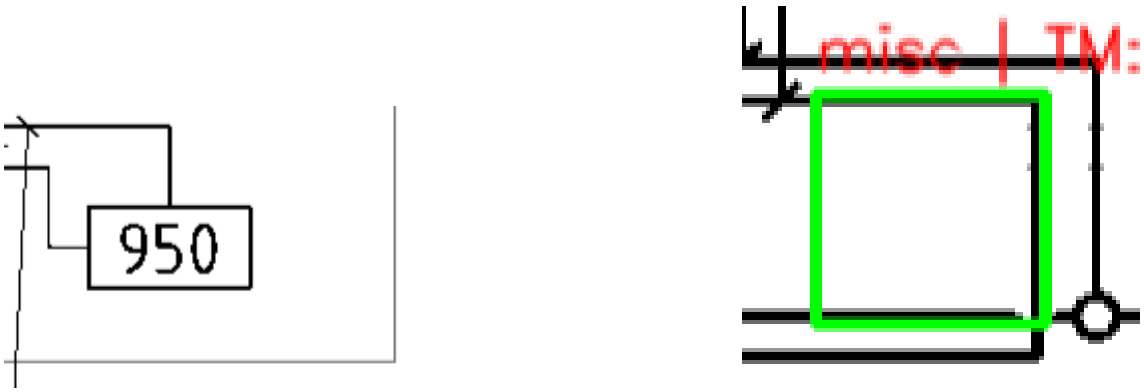


Figure 4.2: Image to the left shows a valid ground truth symbol of the Misc class. Image to the right shows a false positive triggered by structural noise, with three walls made by a wire.

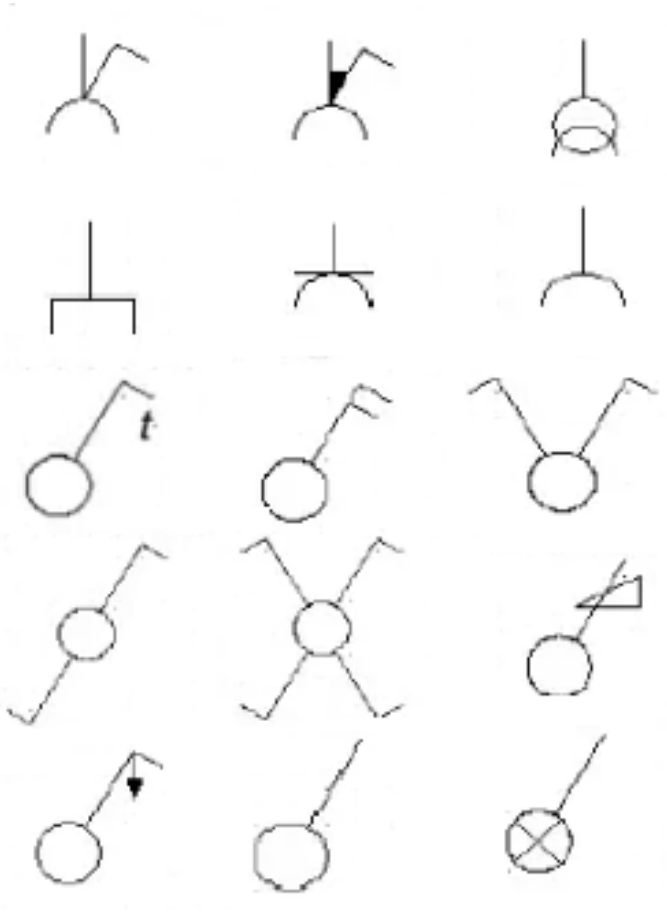


Figure 4.3: Examples of class visual similarity. The top two rows being sockets and the rest switches.

Since there is a very large dispersion of symbol sizes, with the smallest being between 500-600px and the largest being over 15,000px, the impact of scale is evaluated in Figure 4.4. Here the F1-score of each symbol class is mapped against its average bounding box pixel area, calculated as the mean area of all annotated bounding boxes for that class.

A limitation of the method is fine-grained semantic classification. Figure 4.5 shows the eight most confused symbol pairs. The most sever misclassification has a 100% misclassification rate, although only having five instances.

Table 4.1 showcases high levels of false negatives across many of the symbol categories. This is further investigated in Figure 4.6 which shows the 15 symbols most misclassified as background noise. The majority of these undetected symbols are Switch variants, accounting for nine cases, with one of which being incorrectly seen as background noise 100% of instances.

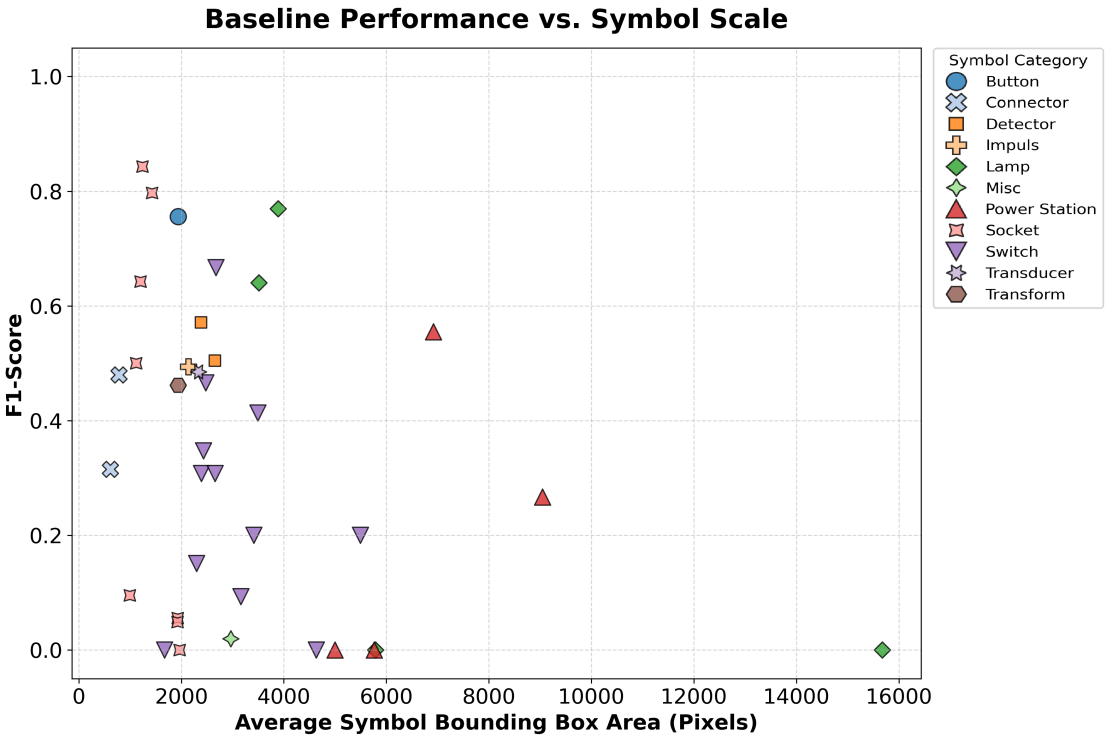


Figure 4.4: F1-score mapped against average bounding box pixel area per symbol.

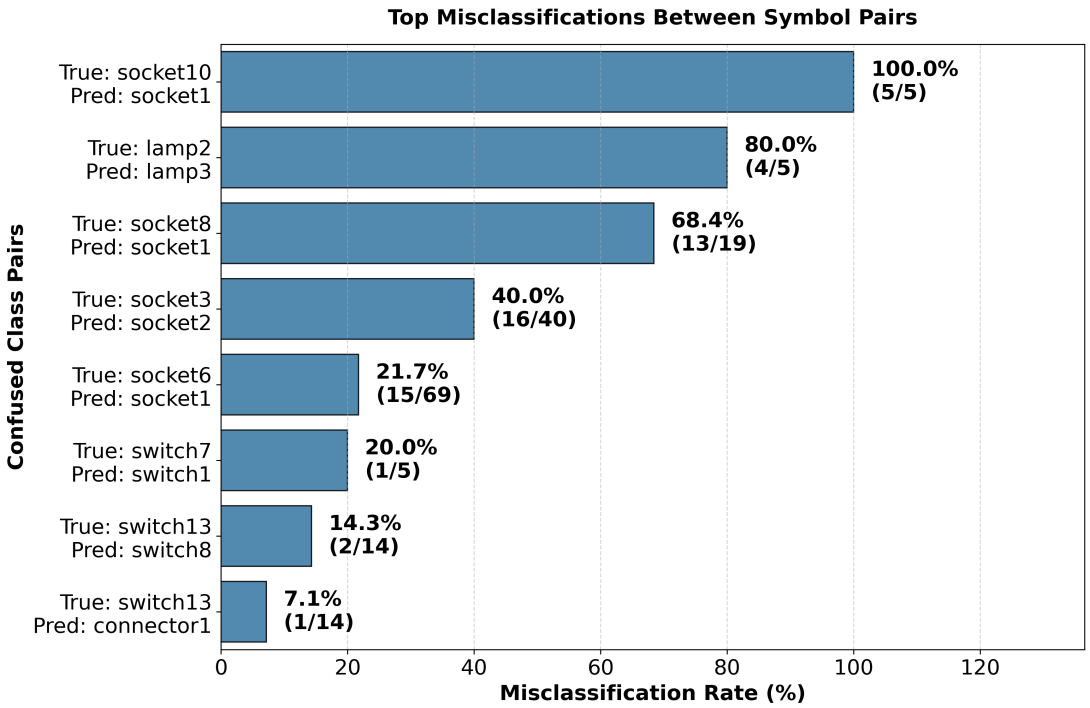


Figure 4.5: Top eight most misclassification between symbol pairs for template matching using sliding window for localization and classification.

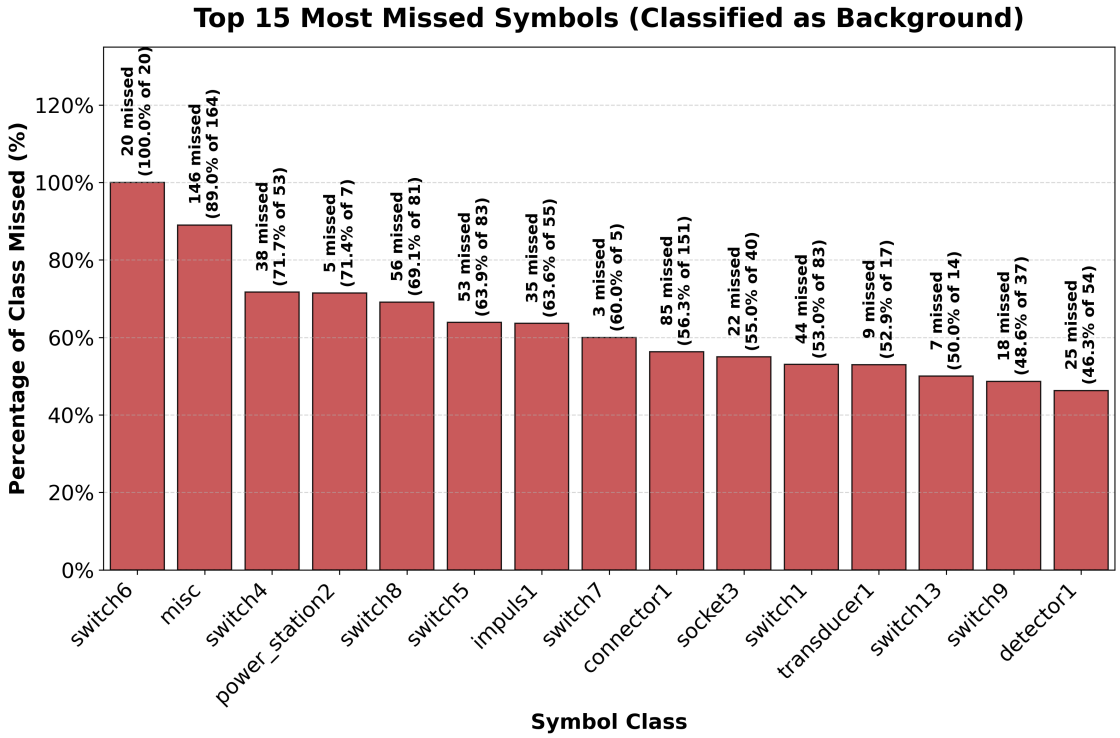


Figure 4.6: Top 15 symbol classes misclassified as background noise (false negatives) for template matching using sliding window for localization and classification.

4.2 Localization Predictions with YOLO, FRCNN and RT-DETRv2

Table 4.2 presents the region of interest detection metrics across all tested models when evaluated on all annotated symbols. The YOLO and RT-DETRv2 models, introduced in section 2.2.1 and 2.3.1 respectively, include results from both their initial pass, as well as their second pass where they were doing inference on the localization predictions and not the entire diagram. The FRCNN, introduced in section 2.2.2 shows results from four different types of passes. RTDETR with a second refinement pass achieves the highest precision (0.2590) and F1-score (0.3567), closely followed by the second pass of RT-DETRv2 and initial pass of YOLOv26m. Although Multi-Scale implementations of FRCNN achieve very high recall (up to 0.8750), they have very low precision and F1-score. Refinement passes marginally improved precision for RTDETR but decreased both precision and recall for all YOLO models.

Similarly, Table 4.3 highlights performance in the same way as Table 4.2, but here only being evaluated on symbols that were included in the associated legend of a diagram. All recall results improved but at the cost of large decrease in precision, with RT-DETRv2 still performing the best on precision and F1-score, and FRCNN with the highest achieved recall of 0.9542.

Table 4.2: Object detection performance on all annotated symbols in the dataset. Models are compared based on 1 pass and 2 pass inference where applicable.

Model	Pass	GT	TP	FP	FN	Precision	Recall	F1-Score
FRCNN	1024	7799	4870	16410	2929	0.2289	0.6244	0.3349
	640	7799	6581	30930	1218	0.1754	0.8438	0.2905
	MultiScale	7799	6741	35662	1058	0.1590	0.8643	0.2686
	MultiScale+WBF	7799	6824	38290	975	0.1513	**0.8750**	0.2579
RT-DETRv2	1	7799	4464	13454	3335	0.2491	0.5724	0.3472
	2	7799	4468	12784	3331	**0.2590**	0.5729	**0.3567**
YOLO11m	1	7799	5567	19895	2232	0.2186	0.7138	0.3347
	2	7799	5576	20635	2223	0.2127	0.7150	0.3279
YOLO26m	1	7799	5455	19144	2344	0.2218	0.6994	0.3367
	2	7799	5473	20395	2326	0.2116	0.7018	0.3251
YOLOv11Freeze	1	7799	6345	23030	1454	0.2160	0.8136	0.3414
	2	7799	6310	23919	1489	0.2087	0.8091	0.3319
YOLOv8	1	7799	6269	22229	1530	0.2200	0.8038	0.3454
	2	7799	6248	22486	1551	0.2174	0.8011	0.3420

Table 4.3: Object detection performance evaluated only on symbols that has been included in the legend of the corresponding diagram.

Model	Pass	GT	TP	FP	FN	Precision	Recall	F1-Score
FRCNN	1024	3754	2590	18690	1164	0.1217	0.6899	0.2069
	640	3754	3506	34005	248	0.0935	0.9339	0.1699
	MultiScale	3754	3555	38848	199	0.0838	0.9470	0.1540
	MultiScale+WBF	3754	3582	41532	172	0.0794	**0.9542**	0.1466
RT-DETRv2	1	3754	2562	15356	1192	0.1430	0.6825	0.2364
	2	3754	2564	14688	1190	**0.1486**	0.6830	**0.2441**
YOLO11m	1	3754	3072	22390	682	0.1207	0.8183	0.2103
	2	3754	3088	23123	666	0.1178	0.8226	0.2061
YOLO26m	1	3754	2997	21602	757	0.1218	0.7983	0.2114
	2	3754	3046	22822	708	0.1178	0.8114	0.2057
YOLOv11Freeze	1	3754	3400	25975	354	0.1157	0.9057	0.2053
	2	3754	3397	26832	357	0.1124	0.9049	0.1999
YOLOv8	1	3754	3368	25130	386	0.1182	0.8972	0.2089
	2	3754	3358	25376	396	0.1169	0.8945	0.2067

Below in Figure 4.7 is a example of how bounding boxes can be placed. The example is from when the FRCNN Multiscale + WBF model is used, showing five predicted bounding boxes, green for TP, red for FP and blue for FN. Three predictions are made successfully with three different levels of IoU being accepted. One is a FP placed over a better made prediction, and one prediction failing the IoU threshold receiving both a FP and a FN.

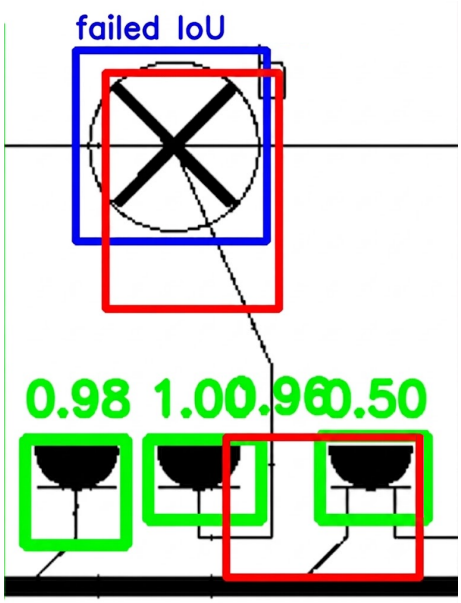


Figure 4.7: Example output of the region of interest prediction models. Here a partial result of the FRCNN Multiscale + WBF is presented, with green boxes indicating TP, red FP, and blue FN.

4.2.1 Predicted Region Sizes

Again, due to the large difference of symbol sizes Figure 4.8 shows the relationship between the median pixel area of each symbol and its corresponding detection recall. Here only the highest performing models of each architecture was chosen, YOLOv11Freeze, FRCNN Multiscale + WBF, and RT-DETRv2. The plot shows that FRCNN and YOLO share a similar performance pattern. Both achieve their highest recall on smaller symbols with the exception of the very smallest, which are connectors at 500-600px, before seeing a decline in performance as symbol size increases. RT-DETRv2 has a much bigger spread then the previous two models for the smaller symbols and has significantly worse performance, but shows a much slower decline as the symbol sizes increases. It also performs better in the 4000-6000px range. Note that the recall decline for localizing larger symbol classes are heavily influenced by the low number of appearances during inference.

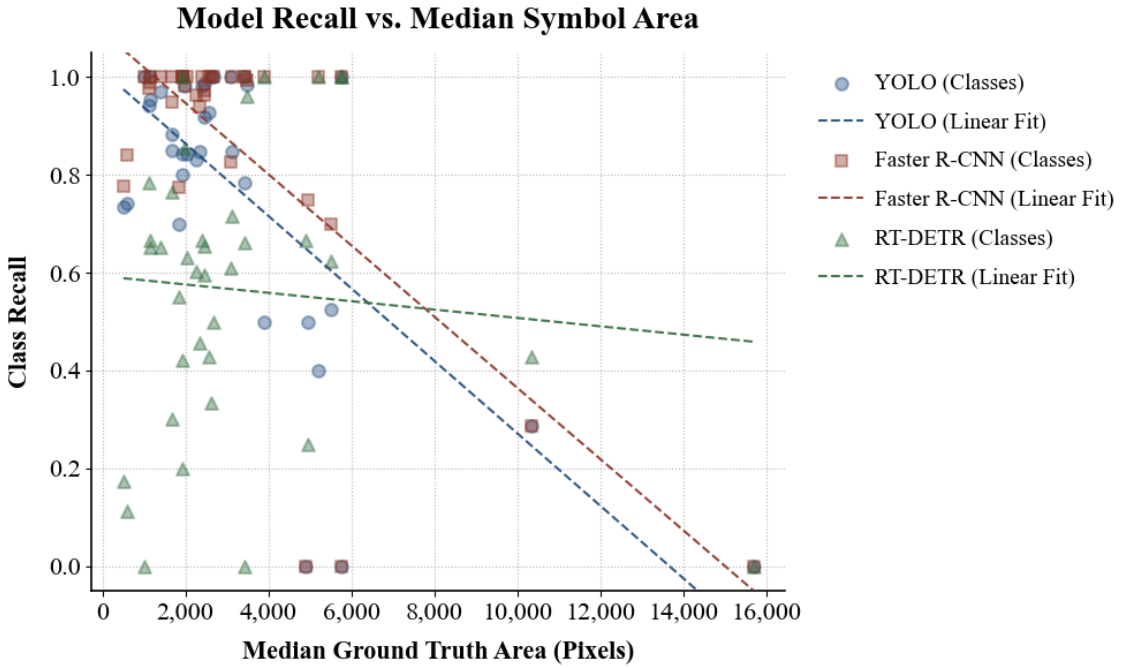


Figure 4.8: Class-wise recall plotted against the median ground truth area (in pixels) for YOLO, FRCNN, and RT-DETRv2. The dashed lines denote linear fit, highlighting the general performance trends as a function of object scale.

The performance trends are shown again in Table 4.4 which aggregates the dataset into discrete size intervals. This breakdown shows that the distribution of symbol sizes are heavily imbalanced and skewed toward smaller scales. The majority of symbols fall below 4,000 pixels in median area, with over 98% of the total instances in the dataset. As the symbol area expands the frequency of available instances drops significantly leaving large objects heavily underrepresented.

Table 4.4: Average recall metrics for YOLO, FRCNN, and RT-DETR aggregated by median symbol area intervals. The instance counts highlight a heavy concentration of targets at smaller scales, with the maximum median class area reaching 15,677 pixels.

Size Interval (px)	Classes	Instances	YOLO Recall	FRCNN Recall	RT-DETR Recall
0–1000	3	360	0.825	0.873	0.095
1000–2000	11	2377	0.902	0.970	0.622
2000–4000	15	951	0.897	0.977	0.588
4000–6000	6	58	0.404	0.575	0.757
6000–16000	2	8	0.143	0.143	0.214

4.3 Classification On Region Proposals

After the localization steps, three different versions of template matching and two different versions of Siamese network were tested. These are denoted as Native TM, Size-Weighted TM which included a added bonus for larger symbols, Multichannel

TM with a multi-channel representation of both the templates and the predicted regions, SM Semi-Hard for being trained with pure semi-hard negative mining and SM Two-Tiered which used a custom batch sampler to pair known hard-negative classes. Because this is a two-stage object detection pipeline there is a performance cap from the first stage. If the localization model fails to predict valid ground truth symbol, the downstream classifiers are completely bypassed affecting end results.

The errors from the first step introduces a minor bottleneck which is stated in Table 4.5 as FN (ROI Miss) while classification false negatives are denoted as FN (Class Error). For all methods, a constant penalty of 172 symbols is included due to misses in localization. Given this, when evaluating the classification, the template matching show better results almost across all methods and metrics. The size weighted variant performed the best with a precision of 0.788 and a recall of 0.688 with the NativeTM following close behind. Although the Siamese network methods were close in recall, they performed significantly worse on precision with magnitudes larger FPs. The SM Two-Tiered method outscores SM Semi-hard in all metrics with a 0.1 higher score in precision and 0.06 higher score in recall.

Table 4.5: Object classification performance on the Legend Symbols dataset using proposed regions of interest from the FRCNN Multiscale + WBF stage.

Classification Method	TP	FP	Total FN	FN (ROI Miss)	FN (Class Error)	Precision	Recall	F1-Score
Native TM	2560	696	1194	172	1022	0.786	0.682	0.730
Size-Weighted TM	2582	696	1172	172	1000	0.788	0.688	0.734
Multichannel TM	2117	656	1637	172	1465	0.763	0.564	0.649
SM Semi-Hard	2496	7613	1258	172	1086	0.247	0.665	0.360
SM Two-Tiered	2519	5815	1235	172	1063	0.302	0.671	0.417

Further investigation on the two top-performing variants, Size-Weighted TM and the Two-Tiered Siamese network, were made and are showcased below. Although having similar recall, the two variants show slightly different trade-offs at different aspects. The top 15 most missed symbols, seen as background noise, as well as the top 10 most common pairwise misclassification for template matching is shown in Figures 4.9 and 4.10. The two Figures show results that are very similar to the previously shown localization and classification results for the sliding-window template matching method found in Figures 4.5 and 4.6. In both cases, switches account for eight of the 15 most frequently missed symbols, and the four most commonly misclassified class pairs are the same.

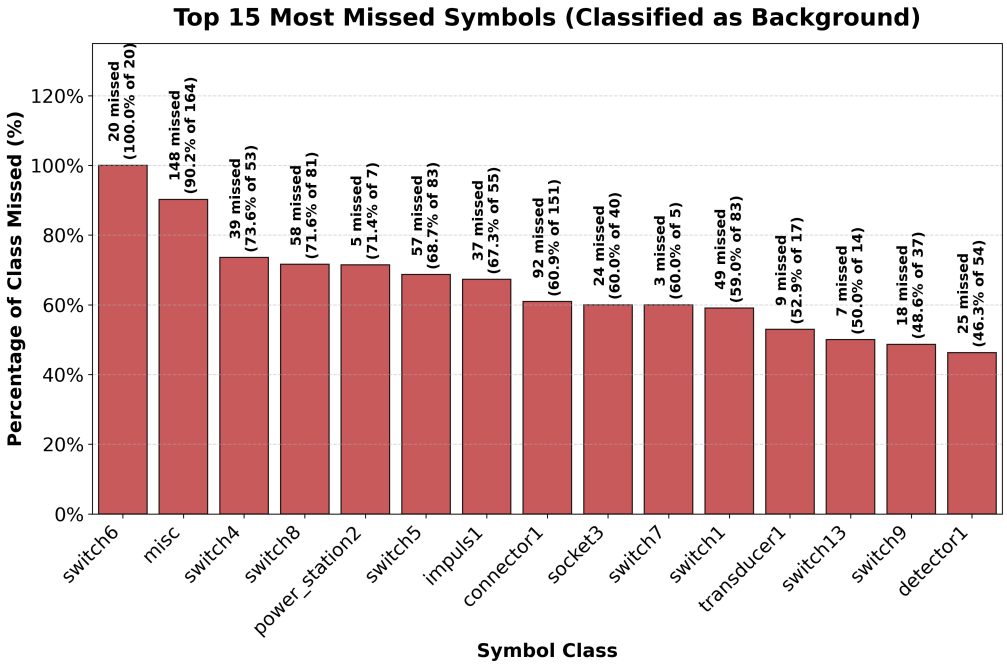


Figure 4.9: Top 15 symbol classes misclassified as background noise (false negatives) for classification with template matching.

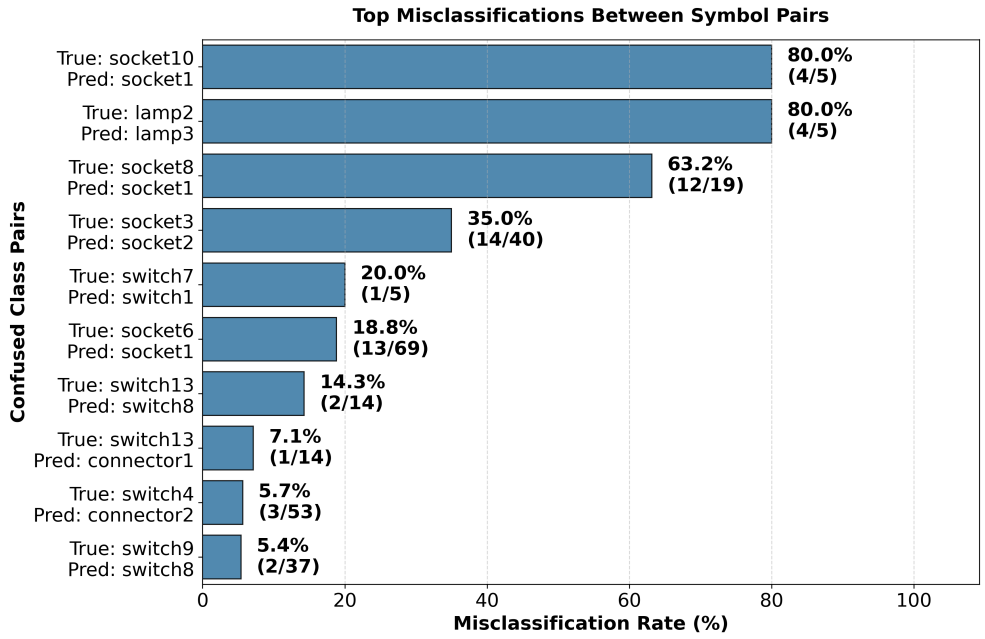


Figure 4.10: Top 10 most misclassifications between symbol pairs when using Size-Weighted TM.

The Two-Tiered Siamese network method shows a slightly different error profile, the model misses valid symbols less often compared to template matching as seen in Figure 4.11. The percentage of class missed is also overall significantly lower with

the highest being 60%, but as seen in Table 4.5 above, it also has vastly more FPs and a much lower precision score.

The Two-Tiered Siamese network also makes notably more misclassification as shown in Figure 4.11. The majority of the class pairs in the Figure are sockets and switches. Additionally 'switch12', 'switch1' and 'socket1' is seen being included in many misclassification pairs and not just once.

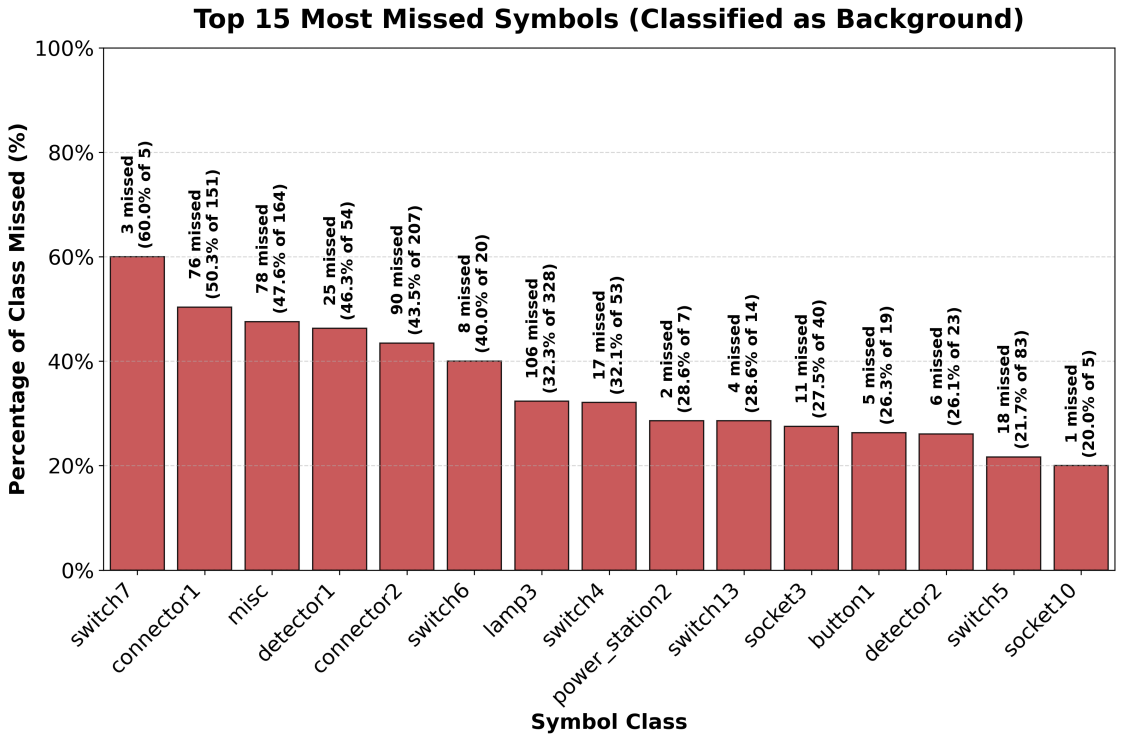


Figure 4.11: Top 15 symbol classes misclassified as background noise (false negatives) for classification using the Two-Tiered Siamese network.

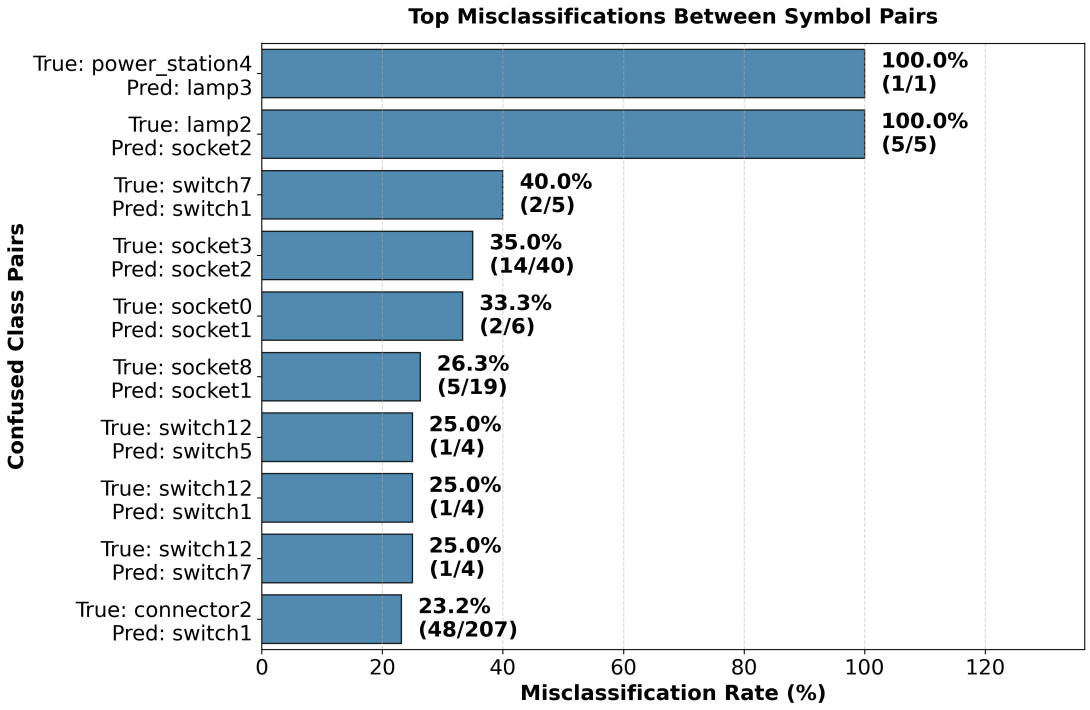


Figure 4.12: Top 10 most misclassifications between symbol pairs using the Two-Tiered Siamese network method.

4.3.1 Classification given Perfect Regions

To establish the upper bound of the classification stage based on the methods used, and completely isolate it from the first stage localization errors, the top-performing methods, Two-Tiered Siamese Network and size-weighted template matching was evaluated using only the perfect, manually annotated ground-truth bounding boxes, removing imperfect crops and majority of background noise inclusion.

As seen in Table 4.6, feeding the network ideal regions gave minimal difference for template matching, but vastly increased the performance for the Siamese network classification. The global precision rises to 0.912 and recall reaches 0.875.

Table 4.6: Global classification performance evaluated on annotated perfect bounding boxes.

Method	TP	FP	FN	Precision	Recall	F1-Score
Size-Weighted TM	2567	997	1187	0.720	0.684	0.702
Two-Tiered SM	3286	316	468	0.912	0.875	0.893

Figure 4.13 shows the most missed symbols seen as background noise for the Siamese network, indicating a significantly lower amount of symbols that were left unclassified with only four symbols having more than 10% of instances missed. A similar trend but to a much lower extent is shown in Figure 4.14 showing the most missed

symbols seen as background noise for template matching. Across the 15 symbols shown in each Figure, the mean missed-symbol percentage decreases by 29.9 percentage points.

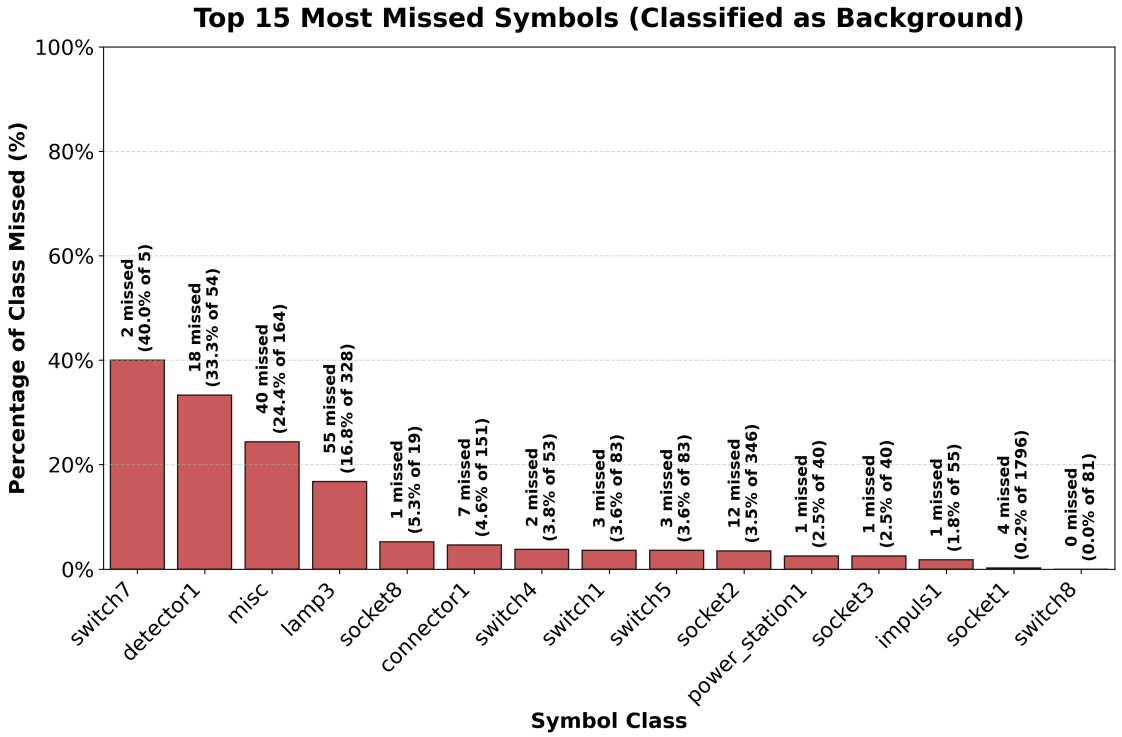


Figure 4.13: Top 15 symbol classes misclassified as background noise (false negatives) for classification using the Two-Tiered Siamese network when provided ideal localization bounding boxes during inference.

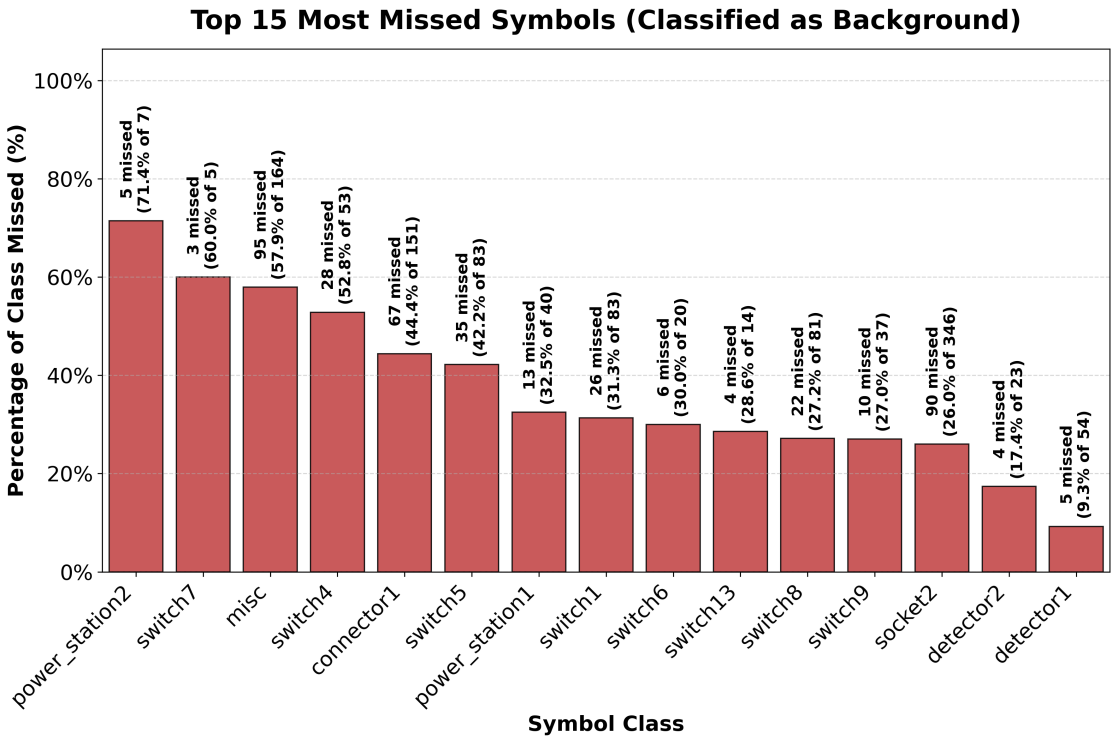


Figure 4.14: Top 15 symbol classes misclassified as background noise (false negatives) for classification using size-weighted template matching when provided ideal localization bounding boxes during inference.

Class misclassification for Siamese network, seen in Figure 4.15 suggests that there is only a slight improvement compared to when receiving non-ideal bounding boxes for inference, with the most noticeable differences being that most of the pairs has changed, even though the percentages of misclassification has not changed much. A vast increase is however seen for template matching as showcased below in Figure 4.16. Template matching previously performed relatively well with only a few symbol pairs with high misclassification rates, but this has increased, with the worst three performing pairs having 100% incorrect classifications.

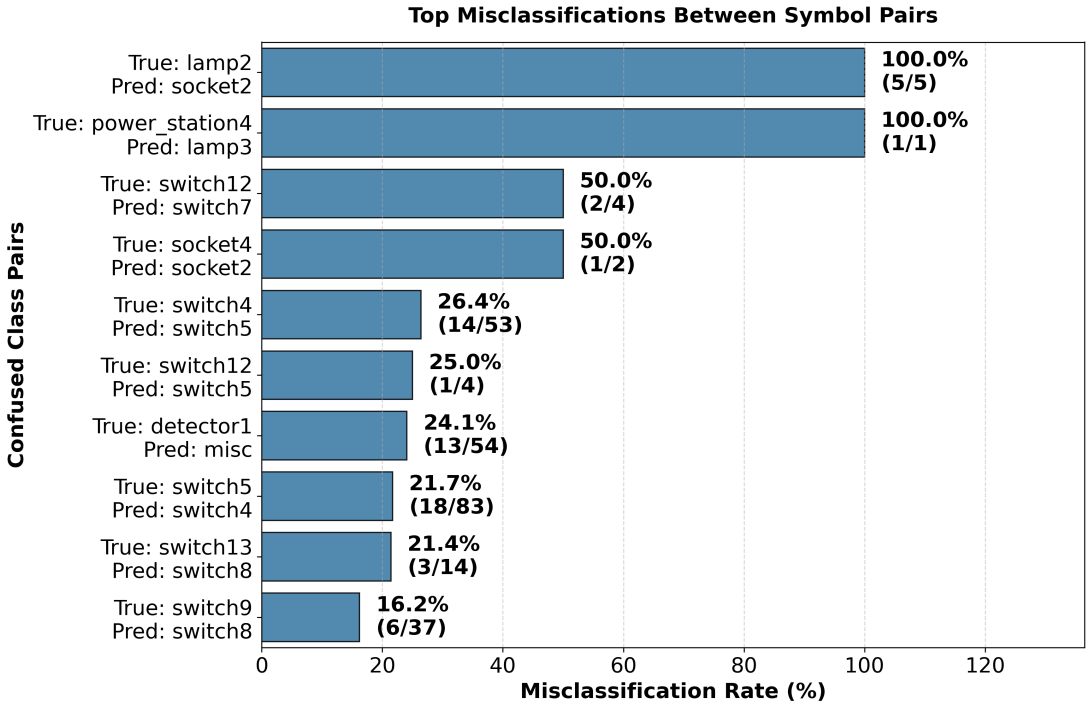


Figure 4.15: Top 10 most misclassifications between symbols for Two-Tiered Siamese network method given ideal localization bounding boxes during inference.

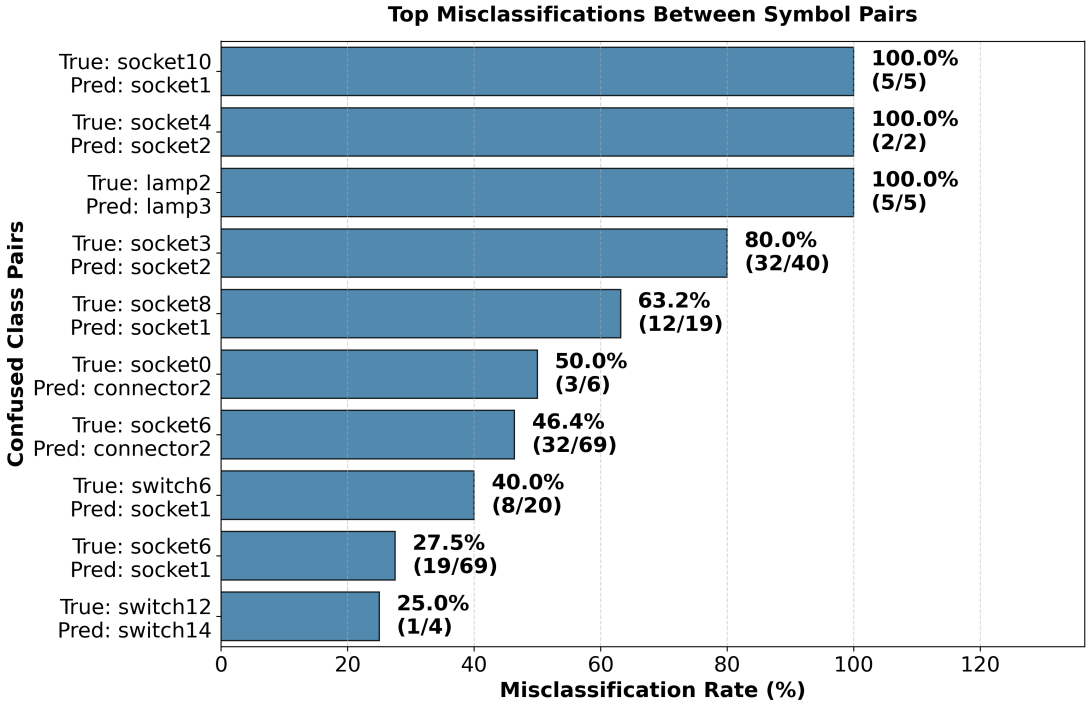


Figure 4.16: Top 10 most misclassifications between symbols for size-weighted template matching given ideal localization bounding boxes during inference.

5
Discussion

The results show that reference-based symbol detection using diagram legends has potential in electrical housing diagrams, but also that the core issue is the difference between the symbols in the legend and the symbols as they appear in the full diagram. The legend provides clean and isolated reference examples while the corresponding symbols in the diagram are placed in dense and visually complex contexts. In the full drawings, symbols are always connected to wires, often times placed in close proximity to other symbols and artifacts, and sometimes preprocessing steps do not clean up the diagrams as intended.

The dataset used throughout the experiments was not balanced and had large variation in both symbol frequency and size. After filtering to only the symbols shown in legends, the dataset contained 3,754 relevant symbol instances across 37 classes, but the distribution is very uneven. Some classes occur only once, while the most common class occurs 1,796 times. The median number of instances per class is only 19 but the mean is 101, which means that many classes only occur a few times, while a few classes dominate. This imbalance makes the overall performance scores heavily skewed on how well the methods perform on the most common symbols, and not as much on performance across all symbol classes.

The dataset is further skewed with respect to symbol size with most instances being small, over 98% of the evaluated symbols having a median area below 4,000 pixels. This makes performance over larger symbols much more sensitive to individual errors. This size imbalance is important due to small symbols being more vulnerable to rasterization, broken strokes, nearby wires and loss of small distinguishing details. At the same time, large symbols are not necessarily easier as they often times comprises of more than one structure and can even be made out of composites of other symbols and are heavily underrepresented in the dataset.

Many symbols can be difficult for both localization and classification, not only due to size, but they can also be visually similar to other symbols, or easily confused with background noise. This is not always represented in the results as can be seen in the sliding-window template matching results where the Socket category gets much better performance than several other categories. This can partly be explained by the visual properties of the socket1 class, which is the symbol with 1,796 occurrences. Some socket symbols such as socket1 are of average size and completely filled in, making them more visually distinct. This can give the template matching method a stronger correlation, but then if such a symbol has a additional protruding or

associated but disconnected line to it, that might be overlooked. This means that some symbols can be separated easier from the background, while being difficult to separate from other classes. For example, a classifier may correctly identify that a region contains a socket-like symbol but still assign the wrong socket class. This explains why strong category-level performance does not necessarily imply that all classifications within the category are reliable.

This however, is not the case for symbols such as switches as they often consist of thin line-based structures that can easily be missed or interpreted as part of the surrounding electrical network. Similarly the Misc class, a rectangular or box-like symbol, can look like ordinary wire structures or other background geometry. This increases the risk of misses, but also creates the separate problem that it may increase false positives, especially for a method like template matching. Additionally, very small symbols, like the connector class, have a symbol area of only 500-600px which is very tiny in a 8192×5787 image, where wires and walls can be thicker than the symbol.

5.1 Region proposals

The region proposal stage is the first step in the two-stage method and defines which parts of the diagram are passed on to the classifier. A missed symbol at this stage cannot be recovered, while a false positive may still be rejected in the classification step. So emphasis was made to achieve high recall for localization, even at cost of lower precision. This is visible in the results, with the highest localization model also having the largest number of false positives, while still being able to produce ideal prediction bounding boxes.

A clear result is that the second inference pass did not provide meaningful improvement. The second pass decreased results for YOLOv11Freeze and YOLOv8 but for YOLO11m and YOLO26m as well as for RT-DETRv2 there was a minute recall gain and did not improve in a way that changes the overall interpretation. This indicates that the refinement strategy was unnecessary. Once the first pass had produced large imperfect regions with more than one symbol in it, reprocessing those crops did not solve the underlying localization problem. In some cases, the second pass may even have removed useful context.

5.1.1 Model comparisons

The newer YOLO versions generally did not perform better than older versions, which is not consistent with the expectation that newer small-object handling would provide better performance. The layer freezing did however perform very well, indicating that keeping parts of the pretrained representation fixed can be very beneficial in this domain. This suggests that the general low-level features learned during pretraining were useful enough to transfer to diagram localization, while full adaptation may not have been necessary or maybe even risk overfitting to the synthetic training

distribution.

For FRCNN, the 640-pixel input performed better than the 1024-pixel input. The multiscale FRCNN method however, improved the recall further by combining detections from both scales, indicating that the two input sizes captured somewhat different symbol instances. This improvement in recall was very small however, and also with a very similar decrease in precision when compared to the single 640-pixel pass. The multiscale setup therefore improves coverage, but with the drawback of adding more FP proposals. The same can be said with WBF replacing NMS.

For RT-DETRv2, the results suggest a different behaviour than the CNN-based models. It performed worse on many of the smaller symbols, but showed stronger behaviour for larger symbols in the 4000–6000 pixel interval. This indicate that the transformer-based architecture benefits more from larger structures and broader spatial context, while struggling when the relevant symbol information is compressed into very small regions which is consistent with the DETR architecture. Since the dataset is so skewed towards smaller symbols, DETR based models would then generally perform worse than their CNN-based counterparts.

5.1.2 Synthetic Data Generation for Localization Models

The localization models were trained on synthetic diagrams made from generic procedurally generated symbols augmented from a list of basic shapes. This was done to make the detector class-agnostic and generalizable as it should learn to find symbol-like regions, not memorize the specific symbols in the evaluation set. The results show that this worked to some extent, since the models were able to transfer from generic shapes to real electrical symbols and achieve high recall, but the drawback being producing many imperfect boxes and false positives. The generated diagrams contained a lot of noise, but the noise was simpler than real diagram structure. Generated noise could be horizontal, vertical or diagonal but extended from one end of the image to the other, real electrical diagrams would however have more lines and a broader range of wire junctions and would connect to symbols, overlap with them and create more ambiguous boundaries. Because of this, the model might learn where symbol is located, but not as well where the symbol ends and the surrounding wiring begins. This can lead to predictions that include too much context, mistakenly cut off parts of the symbol or detect local wire structures as symbols.

The synthetic diagrams also intentionally contained more symbols than a typical 1024×1024 slice of a real diagram. This supports recall, since the model is trained to expect many symbols and actively search for them. The trade-off is lower precision, because ambiguous regions are more likely to be interpreted as potential symbols as shown by the large number of wire junctions being misclassified as the Misc class.

More realistic synthetic data could therefore improve localization, especially by

adding structured wiring, more realistic symbol placement, varied clutter density and connections between symbols and lines. Using more real symbols or highly specific diagram layouts however, could reduce the generalizability of the approach. The aim of this stage is not to train a detector for a fixed symbol list, but to support detection across documents. The synthetic data should better model the context around symbols while still keeping the symbols themselves generic.

5.2 Template matching for localization and classification

Classification on localization predictions using template matching seen in Table 4.5 showed that the native and size-weighted template matching variants had very similar behaviour. Native TM got a F1-score of 0.730 while the Size-Weighted method reached 0.734, making it the best performing template-based classifier, but only by a small margin. Larger symbols can comprise of smaller subsymbols which can be incorrectly classified, so adding a size-based bonus tries to compensate for this making it more likely to give the larger symbol a higher score.

The multichannel template matching variant performed the worst with an F1-score of 0.649 and a much lower recall of 0.564. This shows that the additional channels did not work as intended in practice. A likely reason is that the extra representation just made the matching more sensitive to noise. Small changes in width, surrounding wires or artifacts may affect each channel differently, so while making the comparison more informative, having several representations may have added more ways for otherwise valid symbols to fail the match.

Template matching using sliding window over the entire diagram gave promising results, as seen in Table 4.1. The method reached an overall recall of 0.702, but an F1-score of only 0.519 due to the large number of false positives. The result was highly class-dependent with Socket reached an F1-score of 0.788, while Switch and Misc performed very poorly. This shows that direct matching with legend symbols as templates can work but breaks down when symbols are thin, similar to other classes, or similar to background structures. The method is also computationally heavy since each template must be searched over large high-resolution diagrams and the resulting response maps needs NMS.

5.2.1 Performance Across Variants

One of the more interesting results is that template matching changed very little across the different experiments. The sliding-window method reached an overall F1-score of 0.519 and recall of 0.702, the best performing template matching on predicted regions reached 0.734 and 0.688, and Size-Weighted TM on perfect prediction bounding boxes reached 0.702 and 0.684. The improvement from sliding window to classification on predicted regions is expected, since the first localization step re-

moves almost all of the search problem. Yet, because perfect bounding boxes did not improve template matching, it suggests that the bottleneck was not localization, but rather the template comparisons. Even when the crop is correct, the template comparison itself still struggles with the difference between the clean legend symbol and the diagram instance.

Template matching using the normalized correlation coefficient method partly explains why this behaviour is prevalent. Since the score is based on mean-subtracted normalized correlation, the method is heavily dependent on the relative spatial arrangement of dark and light pixels. A correct bounding box therefore does not guarantee a high score if the symbol instance differs in stroke width, scale, connected wiring, rotation or internal proportions. Perfect localization removes the search problem, but it does not remove the appearance gap between the legend crop and the diagram crop. This is further explained with having five degree increments in rotation during inference. These incremental steps may be too large for templates to align accurately with certain symbols. A solution can be creating additional templates at finer rotational increments, but having too fine grained increments would make the method cease to be viable due to the immense increase in templates to iterate through.

The error distribution also shows that template matching more often rejects symbols as background than it does confusing them with another class. This is expected because the prediction crops often contains noise. If a proposed region includes too much wiring, cuts off part of the symbol or has a symbol at a slightly different scale than the legend reference, all template scores may become too weak to pass the classification threshold. This changes when provided perfect localization bounding boxes, since every crop already has a valid symbol minimising misclassifying symbols as background, making class-to-class errors more visible.

Looking at the similar scores across the three scenarios suggest that template matching reaches a performance ceiling in this task. Better localization helps remove some false positives from the sliding-window detection, but it does not solve the limitations of direct pixel-pattern comparison. The method performs well and has promising results, but if tight and consistent symbol crops could be produced, alternative shape-based methods such as chamfer matching would be more suited, since they compare contour structure rather than the full normalized intensity pattern.

5.3 Siamese network for classification

The Siamese classifier results suggest that the sampling strategy mattered. Semi-hard negative mining forces the network to learn from non-trivial examples, and the smart batch handler improved this further by placing pre-determined visually similar symbols in the same batch. This is important because many random negative pairs are too easy and contribute little to the embedding structure. Forcing difficult pairs into the same batch likely made the model spend more training effort on the class boundaries that are most relevant in the real task. Despite this, the Siamese

classifier performed worse than template matching on predicted regions, the main issue being precision. Unlike template matching, its errors were more often due to class misclassification than to symbols being interpreted as background, meaning that it often saw a valid symbol-like structure but placed it near the wrong reference in embedding space.

The perfect prediction bounding boxes result are especially important. When given ideal symbol crops, the Siamese classifier improved significantly and reached the strongest classification performance among the tested methods with a precision of 0.912 and recall of 0.875. This shows that the learned representation itself was useful. The poor proposed-region result indicates that it was highly dependent on crop quality. With clean inputs, the model could separate the symbols well but with noisy region proposals, the embedding became too sensitive to context and scale variation.

5.3.1 Synthetic Data Generation

The difference between proposed-region and perfect-box performance is likely connected to training using only synthetic data. The Siamese model was trained on augmented legend crops and synthetic noise. The training data appears to have been sufficient for learning symbol identity under clean conditions, but not enough for the noisy crops produced by the localization stage.

This is a limitation of the synthetic setup, not only on the Siamese method. The model was trained without real annotated diagram crops due to the limited dataset. Larger symbol-context crops from real diagrams, even in small numbers, could make a large difference in model performance. The taken crops would teach the model how symbols appear when connected to wires and surrounded by realistic noise. These crops could then be augmented to create many more instances with changes in scale, rotation, adding noise, removing noise and dilation, creating many training and test cases from a single crop.

5.3.2 Limitations

The biggest limitation in this work is lack of annotated real diagram data. Only a small number of diagrams are annotated and are therefore reserved for evaluation and not training. No model is trained directly on real diagrams. All training data is synthetically generated, so all methods must handle the gap between the synthetic training examples and appearance of symbols in real drawings.

Another limitation is the available computational resources. Experiments are made using an NVIDIA A30 GPU on a shared computing cluster. This allows training and evaluation of modern deep learning models but sets a limit on the number of large-scale experiments, hyperparameter tuning, and repeated training runs that can be done.

The scope is limited to symbol detection and classification based on visual symbol appearance. The work does not attempt to fully interpret the functions of the diagram, reconstruct complete circuits, or extract textual and semantic information.

5.4 Future Works

The promising results in this thesis show that reference-based symbol detection is feasible, but also that the quality of both region proposals and synthetic training data has a large influence on the final performance. The most important direction for future work is therefore the development and procurement of higher-quality data. In this work, the synthetic diagrams were designed to be general and class-agnostic, but the generated structure was still simpler than real electrical housing diagrams. Future synthetic data generation should include more realistic wiring patterns, structured symbol placement, varied symbol density and more realistic connections between symbols and surrounding structures. This would allow the models to learn not only what symbol-like objects look like, but also how they appear when embedded in realistic diagram contexts.

A second important improvement would be increasing the amount of annotated real data. Although the goal of the thesis was to reduce dependence on manually labelled data, even a small amount of real annotated crops could likely improve robustness. Real crops containing symbols together with nearby wires and background noise would add a complexity that is difficult to simulate synthetically.

The results suggest further focus on quality of predicted localization bounding boxes. Siamese network perform much better when given clean and well-localized crops, so further work should investigate methods for producing tighter and more consistent symbol proposals. This could include improvements with synthetic training data, segmentation-based refinement, post-processing of predicted regions, or detectors trained specifically to separate the symbol from attached wiring. One interesting method would be to train a model to detect and remove all non-symbol lines. This would lead to only having symbols left, vastly decreasing the difficulty of localization.

With more and higher-quality data, future work should also evaluate stronger end-to-end localization and classification models. Larger detectors such as Open-World Localization (OWL)-based models or DIstillation with NO labels (DINO)-based detectors could be tested to determine whether stronger pretrained visual representations improve generalization to dense diagram symbols. Vision-language models could also be investigated as part of the process, either for validating detections, interpreting legend entries or helping with the more ambiguous classifications. Segmentation-based models such as Segment Anything Model (SAM) may also be useful for refining proposed regions, especially when bounding boxes include attached a lot of noise. At the same time, simpler but more efficient detectors, such as newer YOLO variants, should not be ignored, since the task still requires processing very large diagrams where speed and scalability are important. It would also be

possible to add a attention mechanism into the detector architecture, for example, Convolutional Block Attention Module (CBAM) could be added to a YOLO- or FRCNN-based model.

Finally, the methods should be evaluated on other types of industrial schematics, such as P&IDs for a true test of generalizability. Testing the same two stage reference-based method across different diagram domains would show whether the combination of synthetic localization, legend-based references and the general shape-based matching can generalize to broader technical drawing interpretation tasks.

6
Conclusion

This thesis investigates reference-based symbol detection in electrical housing diagrams, using diagram legends containing symbol examples and information. The results are promising, showcasing that isolated legend symbols can be used as the reference basis for both localization and classification. Throughout the experiments the legend provided the symbol references, allowing them to be identified without training on any real diagrams. This shows that legend-based references can provide enough information to support symbol detection in settings where annotated real data is very limited.

The thesis also gives the argument that augmentation of ideal symbol examples can be useful when having limited amounts of data. Synthetically generated data made it possible to train both localization models and Siamese network classifiers without relying on large labelled datasets. However, the experiments also show that the complexity of this synthetic data is important as the synthetic diagrams and augmented crops were sufficient for learning general symbol-like structures and clean symbol identity, but they did not fully recreate how symbols appear in real data with complex wire connections and surrounded by complex noise.

Template matching performance gave promising and consistent results across different scenarios indicating a reached practical ceiling due to its sensitivity to changes in scale, stroke width, rotation and connected wiring. The Siamese network showed stronger potential when given clean and well-localized symbol crops, but its performance on predicted regions was limited by the noisy and large prediction bounding boxes. This shows that improvements does not have to be only on classification, but that focus should also be put on producing better region proposals that isolate the symbols.

Overall, the proposed methods demonstrate that reference-guided symbol detection is a feasible and flexible approach for electrical housing diagrams. The separation between localization and classification is especially promising, since it allows the system to use generic symbol detectors together with diagram-specific legend references for classification. At the same time, further improvements are needed before the approach can be considered robust and used in real world industry applications. Future works should focus on more realistic synthetic data, a small amount of annotated real symbol-context crops and better localization. Despite this, the methods presented in this thesis provide a good foundation for being able to reduce manual work in the digitization of electrical housing diagrams.

Bibliography

[1] ManuBGZ. Transformador de visión (ViT - en galego). Wikimedia Commons, August 2024. CC0 1.0 Universal Public Domain Dedication.

[2] Xiaokong Ma. The triplet loss in cosine similarity. ResearchGate, August 2020. Licensed under CC BY-SA 4.0, available at `https://creativecommons.org/licenses/by-sa/4.0/`.

[3] Riksbyggens BRF Björken i Norrköping. Slottsgatan 121 vån1 kraft bel tele. `https://bjorken.se`, 2026. Accessed: May 11, 2026.

[4] J2Elteknik. [relationshandlingar]. `https://www.j2elteknik.se/relationshandlingar/`, 2026. Accessed: May 26, 2026.

[5] Laura Jamieson, Carlos Francisco Moreno-García, and Eyad Elyan. A review of deep learning methods for digitisation of complex documents and engineering diagrams. *Artificial Intelligence Review*, 57(6):136, May 2024.

[6] Antonia Hain, Simon Gölzhäuser, Nicolas Réhault, Thomas Brox, and Matthias Demant. *Legend-Informed Symbol Recognition in Engineering Diagrams with Self-supervised Learning*, volume 16020, page 403–421. Springer Berlin Heidelberg, Berlin, Heidelberg, 2026.

[7] Zixuan Zhu, Qingxia Li, and Zhenyu Fang. Yolo-based component detection system for electrical diagrams. In *2024 11th International Conference on Dependable Systems and Their Applications (DSA)*, page 365–373, Taicang, Suzhou, China, November 2024. IEEE.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 779–788, Las Vegas, NV, USA, 2016. IEEE.

[9] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, Jan 1981.

[10] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8:679 – 698, 12 1986.

[11] Henk J. A. M. Heijmans. Mathematical morphology: A modern approach in image processing based on algebra and geometry. *Siam Review - SIAM REV*, 37:1–36, 03 1995.

[12] Maximilian F. Theisen, Kenji Nishizaki Flores, Lukas Schulze Balhorn, and Artur M. Schweidtmann. Digitization of chemical process flow diagrams using deep convolutional neural networks. *Digital Chemical Engineering*, 6:100072, Mar 2023.

[13] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 1:886–893, Jul 2005.

[14] J Matas, O Chum, M Urban, and T Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, Sep 2004.

[15] P F Felzenszwalb, R B Girshick, D McAllester, and D Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, Sep 2010.

[16] J.P. Lewis. Fast normalized cross-correlation. *Ind. Light Magic*, 10, 10 2001.

[17] Jonathan Weber and Salvatore Tabbone. Symbol spotting for technical documents : An efficient template-matching approach. 11 2012.

[18] OpenCV. Introduction - opencv 4.12.0 documentation, 2026. Accessed: May 12, 2026.

[19] OpenCV Developers. *Template Matching*. OpenCV, 2026. Accessed April 2026.

[20] OpenCV Developers. *Template Match Modes*. OpenCV, 2026. Accessed April 2026.

[21] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: two new techniques for image matching. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'77, page 659–663, San Francisco, CA, USA, 1977. Morgan Kaufmann Publishers Inc.

[22] G. Borgefors. Hierarchical chamfer matching: a parametric edge matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):849–865, November 1988.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[25] Shuo Liu and Zheng Liu. Multi-channel cnn-based object detection for enhanced situation awareness. 2017.

[26] Leo Thomas Ramos and Angel D. Sappa. A decade of you only look once (yolo) for object detection: A review. 2025.

[27] Ross Girshick. Fast r-cnn. 2015.

[28] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 936–944, Honolulu, HI, 2017. IEEE.

[29] Glenn Jocher and Jing Qiu. Ultralytics yolo26, 2026.

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. *Microsoft COCO: Common Objects in Context*, volume 8693, page 740–755. Springer International Publishing, Cham, 2014.

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, page 248–255, Miami, FL, 2009. IEEE.

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. 2015.

[33] Aibo Song, Huang Kun, Bowen Peng, Rui Chen, Kun Zhao, Jingyi Qiu, and Kaixuan Wang. Edrs: an automatic system to recognize electrical drawings. In *2021 China Automation Congress (CAC)*, page 5438–5443, Beijing, China, October 2021. IEEE.

[34] Maximilian F. Theisen, Kenji Nishizaki Flores, Lukas Schulze Balhorn, and Artur M. Schweidtmann. Digitization of chemical process flow diagrams using deep convolutional neural networks. *Digital Chemical Engineering*, 6:100072, March 2023.

[35] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM Computing Surveys*, 54(10s):1–41, January 2022.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. (arXiv:1706.03762), August 2023. arXiv:1706.03762.

[37] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2020.

[38] Md Reshad Ul Hoque, Xin Wei, Muntabir Hasan Choudhury, Kehinde Ajayi, Martin Gryder, Jian Wu, and Diane Oyen. Segmenting technical drawing figures in us patents. In *Proceedings of the Workshop on Scientific Document Understanding (SDU 2022)*, volume 3164 of *CEUR Workshop Proceedings*, pages 1–6. CEUR-WS.org, 2022.

[39] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. 2020.

[40] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955.

[41] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. 2020.

[42] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. 2023.

[43] Wenyu Lv, Yian Zhao, Qinyao Chang, Kui Huang, Guanzhong Wang, and Yi Liu. Rt-detrv2: Improved baseline with bag-of-freebies for real-time detection transformer. 2024.

[44] Congying Wu, Haozheng Yu, Yu Liu, and Chao Gong. Towards reliable power grid modeling from drawings: A review of intelligent understanding, topology inference, and model generation. *Machines*, 14(4):371, March 2026.

[45] Ikenna Ekeke, Carlos Francisco Moreno-García, and Eyad Elyan. Attention-based framework for automated symbol recognition and wiring design in electrical diagrams. *Applied Artificial Intelligence*, 39(1):2548834, December 2025.

[46] Gregory R. Koch. Siamese neural networks for one-shot image recognition. 2015.

[47] Mohit Gupta, Chialing Wei, and Thomas Czerniawski. Semi-supervised symbol detection for piping and instrumentation drawings. *Automation in Construction*, 159:105260, March 2024.

[48] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815–823, Boston, MA, USA, 2015. IEEE.

[49] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. 2016.

[50] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, January 2021.

[51]

[52] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? 2014.

[53] Andrzej D. Dobrzycki, Ana M. Bernardos, and José R. Casar. An analysis of layer-freezing strategies for enhanced transfer learning in yolo architectures. *Mathematics*, 13(15):2539, August 2025.

[54] Matthew Z. Wong, Kiyohito Kunii, Max Baylis, Wai Hong Ong, Pavel Kroupa, and Swen Koller. Synthetic dataset generation for object-to-model deep learning in industrial applications. *PeerJ Computer Science*, 5:e222, October 2019.

[55] Bram Vanherle, Steven Moonen, Frank Van Reeth, and Nick Michiels. Analysis of training object detection models with synthetic data. 2022.

[56] Michał Staniszewski, Aleksander Kempski, Michał Marczyk, Marek Socha, Paweł Foszner, Mateusz Cebula, Agnieszka Labus, Michał Cogiel, and Dominik Golba. Searching for the ideal recipe for preparing synthetic data in the multi-object detection problem. *Applied Sciences*, 15(1):354, January 2025.

[57] Roman Solovyev, Weimin Wang, and Tatiana Gabruseva. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, 107:104117, March 2021.

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY