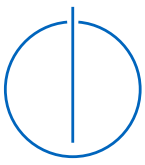
SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Towards Open-World Understanding of Construction Tools and Equipment: A Two-Stage Framework Combining Fast Detection and Vision-Language Reasoning

Sofiia Storozhenko



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

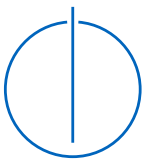
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Towards Open-World Understanding of Construction Tools and Equipment: A Two-Stage Framework Combining Fast Detection and Vision-Language Reasoning

Open-World-Verständnis von Baugeräten und Bauausrüstung: Ein zweistufiger Ansatz zur Kombination schneller Objekterkennung mit visuell-sprachlichem Reasoning

Author:	Sofiia Storozhenko
Supervisor:	Prof. Dr.-Ing. habil. A. C. Knoll
Advisors:	P. Petropoulakis and Prof. Dr.-Ing. A. Borrmann
Submission Date:	March 16, 2026



I confirm that this bachelor's thesis is my own work, and that I have documented all sources and materials used.

Munich, March 16, 2026 Sofiia Storozhenko

Abstract

Construction sites are highly dynamic environments where misplaced tools and equipment pose risks to both autonomous robots and human workers. Reliable real-time perception is therefore essential for safe robotic operation. However, traditional object detection systems are limited to predefined categories and struggle to generalize to unseen tool variants commonly found on construction sites.

This thesis proposes a two-stage framework for open-world understanding of construction tools and equipment. In the first stage, a real-time YOLOv12 detector was trained using a single-class configuration to accurately localize tools under challenging floor conditions. To support training, a synthetic dataset generation pipeline was developed that combines cropped tool images with diverse background scenes that represent a robot's downward-facing perspective. The trained detector achieved very high performance, reaching an mAP@50 of approximately 99.5% and an mAP@50–95 of about 98.7%, with precision and recall both reaching 100% on the evaluation dataset.

In the second stage, the detected regions were analyzed using a Vision-Language Model (VLM) to provide semantic interpretation and hazard reasoning. By combining fast spatial localization with high-level multimodal reasoning, the system moves beyond rigid class labels toward flexible tool identification and contextual understanding. Experimental evaluation showed that the model correctly interpreted most tools and produced meaningful safety descriptions.

The proposed approach demonstrates that integrating real-time detection with vision–language reasoning provides a scalable, adaptable solution for robotic perception in dynamic construction environments, supporting safer, more reliable autonomous robot operation.

Contents

1 Introduction

Construction sites are undergoing a significant transformation. The industry is increasingly exploring the integration of robotics to address persistent challenges such as labor shortages and safety risks. Recent advances in robotics, artificial intelligence, and automation offer considerable potential to enhance productivity, precision, and safety in such dynamic environments [1].

Robotic systems must navigate obstacles, monitor sites, and interact with tools and materials on construction sites. Thus, a robot's ability to detect, classify, and localize objects is essential for making reliable decisions. However, applying object detection in this domain remains difficult [2]. Construction sites are visually complex and constantly changing environments, characterized by cluttered scenes, varying lighting conditions, and a wide variety of tools, equipment, and materials. Furthermore, labeled images of construction sites are scarce, particularly from a robot's perspective. This shortage of representative training data makes it challenging to develop robust object detection and classification systems that perform well in real-world conditions.

Another challenge is detecting small objects. They often contain only limited spatial and contextual information, which makes them considerably harder to detect [3]. Nevertheless, these small tools and equipment lying on the ground may either damage the robot or be damaged by the robotic system. Therefore, accurate real-time detection is essential to prevent accidents during robotic navigation.

To overcome the limited availability of labeled construction-site data, data augmentation can be applied. Synthetic augmentation, in particular, can increase the diversity of training samples by introducing variations in object placement, scale, orientation, and background. Enriching the dataset with a wider range of visual conditions can help improve model robustness and reduce overfitting to limited scene configurations.

Given the need for fast and reliable detection in dynamic environments, real-time object detectors such as You Only Look Once (YOLO) [4] are particularly suitable. YOLO is widely used because it combines high detection speed with strong localization performance. However, such models still primarily rely on predefined classes and may offer limited semantic understanding.

For the reasons mentioned above, this thesis proposes a two-stage framework combining fast detection and vision-language reasoning for construction tools and equipment. In the first stage, YOLO is used to detect and localize relevant objects. In the second

stage, the detected image regions are cropped and processed by a vision-language model, which generates semantic descriptions of the identified tools and equipment. These descriptions are further used to support robot-oriented instructions, thereby extending the system beyond conventional object detection toward a more context-aware understanding of construction-site environments.

Accordingly, the aim of this thesis is to investigate how the proposed two-stage framework can achieve high detection and semantic understanding of safety-critical construction tools in data-scarce and visually complex environments.

To achieve the aforementioned goal, the thesis is structured as follows: Chapter 2 outlines the relevant theoretical background of this study. Chapter 3 reviews related literature, while Chapter 4 describes the methodology and proposed framework. Chapter 5 presents the experimental results and discusses the findings. Chapter 6 concludes the thesis and suggests areas for future research. Finally, Chapter 7 provides supplementary material.

Additionally, the text of this thesis was reviewed using language-assistance tools such as ChatGPT [5] and Grammarly [6] to improve grammar, punctuation, and overall readability. These tools were used solely for language editing purposes. The ideas, methodology, experiments, and substantive content of this thesis were conceived and written by the author.

2 Theoretical Background

This chapter introduces the theoretical foundations relevant to this thesis. Starting with the basic principles of neural networks and backpropagation, it then moves on to convolutional neural networks and object detection approaches. Key concepts that are used in later chapters are also presented, including Intersection over Union, loss functions, data augmentation, and vision-language models.

2.1 Neural Network

Neural Network (NN) is a subset of Machine Learning (ML) that consists of interconnected layers of artificial neurons that process data to recognize complex patterns. Each neuron receives inputs, applies weights to them, and produces a real-valued activation. Neurons in the input layer receive signals from the input data, while neurons in subsequent layers are activated through weighted connections from neurons in the previous layer [7].

Instead of traditional approaches that rely on manually designed feature extractors, neural networks can automatically learn hierarchical representations directly from raw data. This capability enables them to perform complex tasks such as object detection, speech recognition, and natural language processing [8].

Training NNs involves finding suitable weight values that allow the model to produce the desired outputs. This learning process determines how much each connection contributes to the final prediction. In deep learning models, this process may involve many computational stages where each layer transforms the data through non-linear operations [7].

Neural Networks are typically trained using supervised learning and the backpropagation algorithm, which iteratively adjusts the network weights to minimize a loss function [8].

2.2 Backpropagation

As mentioned earlier, backpropagation is used to train NNs by minimizing the difference between predicted and actual outputs. The algorithm determines how the

network's internal parameters (weights) should be adjusted to compute representations in each layer from the activations of the previous layer [8].

The training process works by propagating the prediction error backwards through the network and applying the chain rule of calculus to compute gradients of the loss function with respect to the network parameters. These gradients are then used by optimization methods, such as Stochastic Gradient Descent (SGD), to iteratively update the weights and biases until the overall prediction error is minimized. Under SGD, the parameters are updated using the computed gradients and a learning rate that controls the size of the update step at each iteration [9]. The update rule is defined as:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L\left(\hat{y}, y\right), \tag{2.1}$$

where θ_t represents the model parameters at iteration t, α is the learning rate controlling the step size, with respect to the parameters. L$(\hat{y}, y)$ is a loss function comparing predicted output $\hat{y}$ with the ground-truth value y, while $\nabla_\theta L(\hat{y}, y)$ denotes the gradient of the loss function. The update step moves the parameters in the direction that reduces the prediction error.

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are neural networks specifically designed for difficult image-based pattern recognition tasks. Their architecture is usually constructed by stacking several convolutional layers to learn increasingly complex feature representations, combined with pooling layers for dimensionality reduction. The resulting high-level feature maps are then processed by fully connected layers to generate the final predictions.

Convolutional Neural Networks can directly encode image-specific features, such as edges, textures, and shapes, into its architecture. This approach reduces the number of parameters required to set up the model, thereby minimising computational complexity and the risk of overfitting [10].

2.4 Object Detection

Object detection is a fundamental computer vision task that involves identifying and localizing objects within an image. Object locations are typically represented by rectangular bounding boxes with coordinates (x_1, y_1, x_2, y_2), where (x_1, y_1) refers to the coordinates of the top-left corner and (x_2, y_2) of the bottom-right corner in the image coordinate system.

Modern object detection systems use deep neural networks to predict both object labels and bounding box coordinates directly from images, by learning to extract visual features from input images.

Object detection methods can be divided into two categories: two-stage detectors and one-stage detectors. Two-stage detectors, such as Fast R-CNN [11], first generate region proposals that may contain objects and then classify these regions. Although this approach often achieves high accuracy, it requires multiple processing steps and can be computationally expensive.

In contrast, one-stage detectors perform object localization and classification in a single network pass. Models such as YOLO [4] directly predict bounding boxes and class probabilities from an image. This design significantly improves inference speed, making one-stage detectors well-suited for real-time applications.

2.4.1 Anchor-Based and Anchor-Free Detection

Object detection models often rely on fixed reference templates, called anchor boxes, to predict object locations. In this method, a pre-defined set of bounding boxes with varying sizes and aspect ratios is placed across the image to accommodate different object shapes. The network then predicts offsets relative to these anchors to better align the anchors with the objects.

Although anchor-based methods can achieve high detection accuracy, they introduce additional complexity because the anchor boxes must be carefully designed and adjusted for different datasets.

In contrast, anchor-free models directly predict object locations from spatial positions on the feature map. These methods typically predict distances from a reference point to the four sides of the bounding box. This simplifies the detection pipeline and reduces the need for manual anchor configuration.

2.5 Intersection over Union (IoU)

Intersection over Union is an evaluation metric used to compare the ground truth bounding box to the predicted bounding box to measure the quality of prediction precision. It is defined as the ratio between the area of overlap and the area of union of the two bounding boxes. A higher IoU value indicates a better alignment between the predicted and the ground-truth boxes. The mathematical formulation is defined as follows:

$$IoU = \frac{\left|B_p \cap B_g\right|}{\left|B_p \cup B_g\right|}, \tag{2.2}$$

where B_p represents the predicted bounding box and B_g denotes the ground-truth bounding box. The numerator corresponds to the area of intersection between the two boxes, while the denominator represents the total area covered by both boxes.

2.6 Loss Function

Loss functions measure the difference between an NN's predicted outputs and the true outcome. During training, the model's goal is to minimise this loss to improve its prediction accuracy.

For binary classification tasks, where each input belongs to one of two classes, the Binary Cross-Entropy (BCE) loss is commonly used:

$$L_{BCE} = -\sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right], \tag{2.3}$$

where n denotes the number of training samples, y_i is a ground-truth label (0 or 1) of i, and $\hat{y}_i$ represents the predicted probability (typically the output of a Sigmoid activation function).

For multi-class classification problems, where each sample belongs to exactly one class, the categorical cross-entropy loss is used

$$L = -\sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log \left(\hat{y}_{ik} \right), \tag{2.4}$$

where K is the number of classes, y_{ik} denotes the true label of sample i for class k, and $\hat{y}_{ik}$ represents the predicted probability for that class. The true labels are typically represented using one-hot encoding.

2.7 Data Augmentation

Data Augmentation is a technique that artificially generates new, broader training data from an existing dataset, usually to train new Machine Learning models. A primary theoretical motivation for data augmentation is to overcome data scarcity and reduce the labor-intensive, high cost of manually annotating large-scale datasets [12]. During augmentation, standard mathematical and geometric transformations are applied to existing images to artificially expand the dataset and prevent overfitting, thereby enhancing data diversity. This way, the model is more accurate and can generalize objects more easily.

The most popular transformations are Photometric Distortions (such as color modifications, brightness variations, and contrast changes), Geometric Transformations (such

as image translation, i.e., spatial shifting of the image within the frame), and Mosaic Augmentation (combining four completely different images into a single training sample, which helps the model learn to detect objects in complex, diverse contexts and at smaller scales) [12].

2.8 Vision-Language Models

The focus of VLMs is on integrating the visual and language modalities. By leveraging paired image–text datasets during pre-training, these models learn to align visual content with natural language, making them effective for tasks that require multimodal reasoning [13].

Unlike traditional deep learning models, which require large amounts of annotated, task-specific data, foundation models use large-scale self-supervised (or unsupervised) learning on vast, less organized datasets, such as web-crawled image-text pairs [13].

3 Literature Review

In the following section, two relevant research directions are presented: object detection, with particular focus on single-stage detectors such as You Only Look Once (YOLO), and Vision-Language Models (VLMs). These serve as the basis for the proposed two-stage framework.

3.1 The Evolution of Real-Time Object Detection

You Only Look Once (YOLO) is a series of real-time object detection systems based on convolutional neural networks. The system was first introduced by Joseph Redmon et al. in 2015 [14]. The community developers have since updated and improved it several times, making it one of the most popular object detection frameworks.

The review details how early versions focused on improving localization accuracy and sensitivity to small objects. To improve performance, YOLOv2 introduced the Darknet-19 feature extractor and a passthrough layer to capture fine-grained spatial features [15]. YOLOv3 built on this by formally integrating multi-scale predictions and a Feature Pyramid Network (FPN) [16], to detect objects of any size or scale within an image, improving the identification of smaller targets [17]. The YOLOv4 model has been enhanced with architectural optimization, incorporating the CSPDarknet-53 backbone and a Path Aggregation Network (PANet) neck [18][19]. This has resulted in improved accuracy and reduced inference time by optimizing parameter utilization and gradient flow [14].

The YOLOv5 model marked a significant change for the YOLO family. It used PyTorch [20] as the framework and included automatic anchor-box learning [21]. More recent versions, beginning with YOLOv8, have adopted a template-free approach, directly predicting the centers of objects and their corresponding bounding box dimensions [14]. This change has simplified the detection pipeline, reduced computational overhead during post-processing, improved generalization, particularly for datasets with varied object scales, and laid the foundation for further advancements.

Recent iterations have continued this evolutionary trajectory by focusing on multi-scale robustness, training stability, and efficient gradient routing. The widespread adoption of multi-scale feature aggregation ensures that semantic features from deeper

layers combine with localization-rich features from shallower layers. This is a foundational principle for detecting objects of different sizes. Furthermore, the transition to decoupled detection heads, which separate classification and regression tasks, has resolved representational conflicts and enabled more specialised and stable learning [14]. YOLOv10 built on this by maintaining the modular structure of the backbone, neck, and head while enabling fully end-to-end training without Non Maximum Suppression (NMS) [22]. YOLOv11 then advanced feature extraction by introducing Partial Spatial Attention (PSA) to emphasize spatially relevant regions [23]. This model produces bounding box predictions at three spatial resolutions, corresponding to low-, medium-, and high-level feature maps, enabling it to detect objects of different sizes [14].

The latest generation, YOLOv12, integrates an attention-focused design into the real-time detection pipeline [24]. This aims to utilize powerful representational capacity without sacrificing high inference speed. A notable change is the integration of the Residual Efficient Layer Aggregation Network (R-ELAN), which enhances stability and computational efficiency [24]. Additionally, Area Attention (A^2) has been introduced. This mechanism segments feature maps into uniformly spaced vertical or horizontal areas via simple re-shaping operations [25]. This limits computational cost, making the detection highly suitable for fast inference compared to global or axial attention. These enhancements maximize the accuracy and robustness of YOLOv12 when detecting small tools and equipment in potentially complex ground environments [24].

3.2 YOLOv12 Architecture

For this thesis, one of the most recent YOLO releases, YOLOv12 [24, 26], was chosen. As was mentioned before, a central focus of YOLOv12 is the incorporation of attention-based design into the real-time detection pipeline [25]. While earlier versions relied almost exclusively on convolutional networks, YOLOv12 aims to utilize the powerful representational capacity and global context of attention mechanisms while maintaining inference speed. The approach maximizes both accuracy and robustness by processing a larger area of the image at once.

As shown in Figure 3.1, the YOLOv12 architecture consists of three main components: the backbone for extracting and processing multi-scale features, the neck for combining and refining these features, and the head for generating the final predictions.

3.2.1 The Backbone

The backbone processes the input image and extracts hierarchical visual features. These include basic edges and textures in the early layers and complex object shapes in

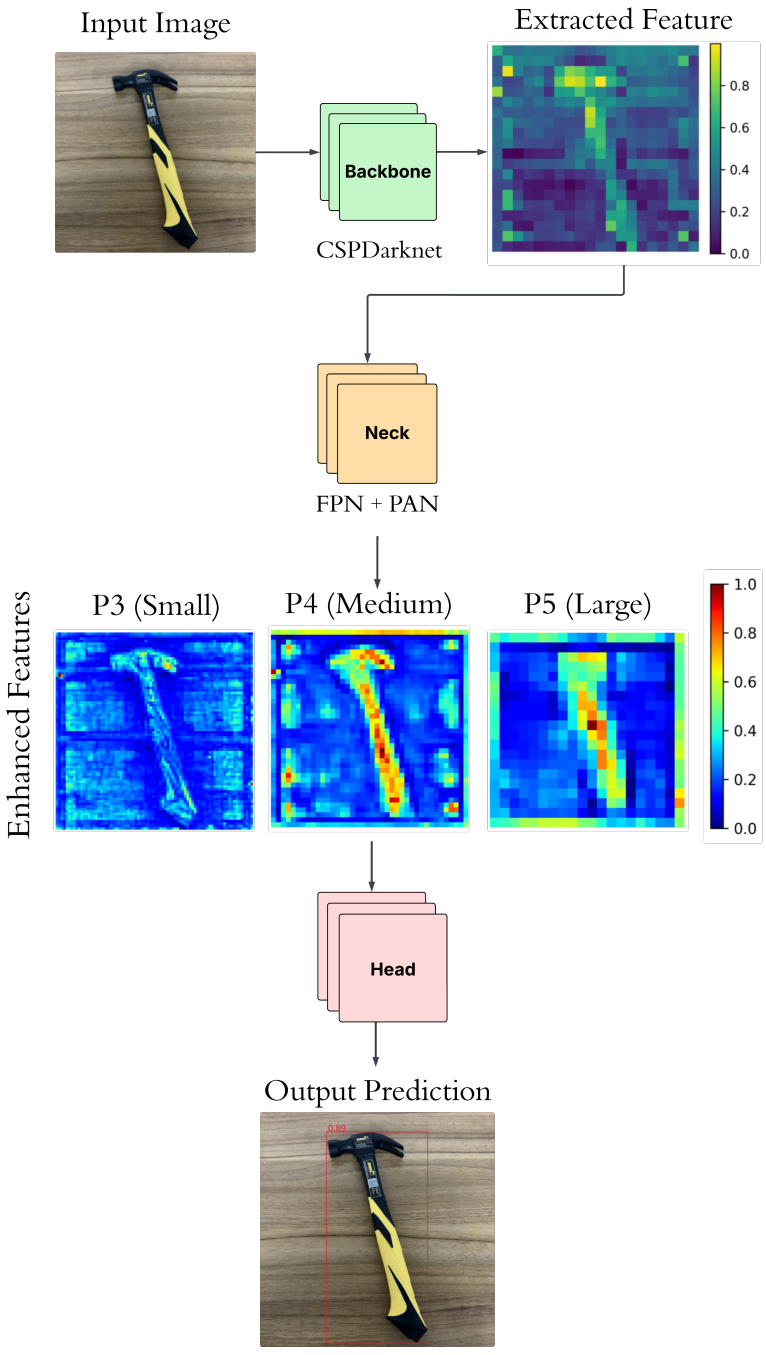


Figure 3.1: Architecture of YOLOv12. Color bars indicate feature activation levels; red (1.0) is considered the highest model focus.

the deeper layers. In the YOLOv12 architecture, the first two stages of the backbone inherit traditional convolutional designs, and the deeper stages transition into an innovative, attention-centric architecture designed to maximize both speed and feature representation. To achieve this, the network relies on two core architectural concepts: Area Attention (A^2) [25], and the R-ELAN [24]. Area Attention segments feature maps into uniformly spaced vertical or horizontal equal segments (typically 4 areas) via simple reshape operations, as shown in Figure 3.2.

These segments are processed independently, reducing computation and improving the model's speed while still capturing contextual relationships across spatial regions. Despite focusing on partitioned regions, the module maintains a large receptive field, enabling the network to capture broader contextual information than axial attention mechanisms.

Figure 3.2: A comparison of representative local and area attention mechanisms [24].

The backbone uses R-ELAN to construct network layers effectively and aggregate features extracted by the Area Attention modules. The idea is to add residual networks [27] (so-called shortcuts) that allow the original information to safely bypass certain complex processing blocks. If a specific layer struggles to process the features, the data can take a shortcut, ensuring the tool's core visual identity is never lost. See Figure 3.3. In the transition stage, a 1×1 convolution is first applied to adjust the feature channels before further processing. The A^2 modules then perform area-based attention by using a simple reshape operation to divide the feature map into equal vertical or horizontal segments, and compute attention within each region. Finally, the feature map is processed through subsequent attention blocks, and their outputs are concatenated and scaled before being combined with the shortcut connection. This helps stabilize training and preserve important feature information, reduces

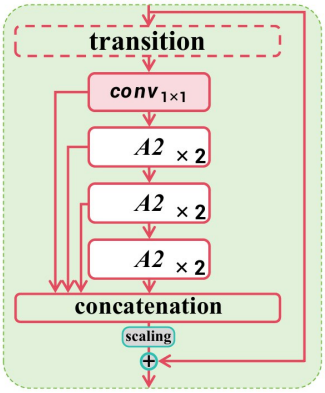


Figure 3.3: R-ELAN [24].

overall computational cost and memory footprint [24].

3.2.2 The Neck

In real-world scenarios, an object can appear at different scales and distances from the robot's sensors. To handle this size variance, the neck employs a multi-scale feature fusion strategy, inspired by PANet [18] and FPN [16].

Rather than relying solely on the deepest layers of a convolutional backbone, which offer a wealth of semantic information but poor spatial resolution, FPN introduces a top-down pathway with lateral connections. This structure combines high-level semantic features with lower-level spatial details to construct a feature pyramid (P3, P4, P5) that enhances detection across different scales. This enables detectors to effectively recognize both small and large objects [16]. P3 is the largest map and stays closest to the original image size, making it ideal for finding small objects. At the same time, P5 is the smallest map, significantly reduced to capture a much broader semantic context, which is better for identifying large objects. P4 sits in the middle, balancing detail and context for medium-sized objects [28]. The specific characteristics of these feature maps are summarized in Table 3.1. As shown in the table, the network progressively reduces spatial resolution to increase semantic depth, enabling each level to specialize in a particular scale of objects.

Table 3.1: Feature Map Specifications in the YOLOv12 Neck [24].

Level	**Resolution** (640×640)	**Stride**	**Receptive Field**
$P3$	$80 \times 80 \times C$	$8\times$	Small/Local
$P4$	$40 \times 40 \times C$	$16\times$	Medium/Regional
$P5$	$20 \times 20 \times C$	$32\times$	Large/Global

*Note: C represents the channel depth (number of filters) at each respective layer.

The Path Aggregation Network (PAN) shares information between these layers. The area attention mechanism further enhances this collaboration by ensuring the model remains focused on the most relevant features at every scale, from the smallest bolt to the largest tool.

3.2.3 The Head

The detection head is responsible for predicting bounding boxes, identifying object categories, and calculating confidence scores.

Bounding Box Detection

The neural network's detection head outputs an array of raw, unnormalized values, called logits, for the four edges of the box: Left, Top, Right, and Bottom.

The probability is distributed over 16 discrete values ($reg\_max = 16$) for each of four directions. The final continuous distance d is calculated using a Softmax layer and a weighted sum: $d = \text{Softmax}(x) \cdot [0, 1, 2, \ldots, 15]$, where x represents the logits. The Softmax function is given by Equation 7.2.

Once the four distances $(d_{\text{left}}, d_{\text{top}}, d_{\text{right}}, d_{\text{bottom}})$ are calculated, the final bounding box coordinates are decoded relative to the anchor point:

$$\begin{aligned} x_1 &= \text{anchor\_point}_x - d_{\text{left}}, \\ x_2 &= \text{anchor\_point}_x + d_{\text{right}}, \\ y_1 &= \text{anchor\_point}_y - d_{\text{top}}, \\ y_2 &= \text{anchor\_point}_y + d_{\text{bottom}}, \end{aligned}$$

where $(\text{anchor\_point}_x, \text{anchor\_point}_y)$ represents the center coordinates of the corresponding anchor cell, a specific location on the feature map that serves as the reference point from which the model predicts the distances to the four sides of the bounding box.

To ensure these predicted boxes align with the ground truth, the model minimizes the IoU Loss:

$$L_{\text{IoU}} = 1 - IoU. \tag{3.1}$$

A detailed definition of the IoU metric can be found in the Theoretical Background in Section 2.5.

Confidence Scoring in YOLOv12

In YOLOv12, the confidence score for a specific class i at inference time is derived by applying the sigmoid activation function to the raw class logit output from the network:

$$\text{confidence}_i = \sigma(z_i), \tag{3.2}$$

where z_i is the raw class logit and σ represents the sigmoid activation function (refer to Equation 7.1 in Supplementary Material).

The model is trained using IoU-aware targets through the Task-Aligned Assignment (TAL) strategy [29]. It aims to align the classification confidence with the localization quality of the predicted bounding box. In other words, the model learns to produce high confidence scores only when two conditions are satisfied simultaneously: the

predicted class is correct, and the bounding box is accurately localized (i.e., has a high Intersection over Union with the ground-truth box). This is achieved using an alignment metric defined as:

$$\text{align\_metric} = s^{\alpha} \cdot \text{IoU}^{\beta}, \tag{3.3}$$

where s denotes the predicted classification score for the correct class. The parameters α and β control the relative importance of classification confidence and localization quality, respectively. Typically, α is set to 0.5, while β is set to 6.0.

The alignment metric is then used to scale the one-hot encoded class labels, producing the final target score used during training. As a result, predictions with both high classification confidence and accurate localization receive stronger supervision, encouraging the model to optimize classification and bounding-box regression simultaneously.

The final target score used for training incorporates this alignment metric to appropriately scale the one-hot encoded class labels:

$$\text{target\_score} = \text{one\_hot}\,(\text{class}) \times \frac{\text{align\_metric} \cdot \text{IoU}}{\max\,(\text{align\_metric}) + \epsilon}, \tag{3.4}$$

where ϵ is a tiny constant added to prevent "division by zero" errors if no anchors align well, and one_hot(class) is a vector where the index of the ground truth category is 1 and all other categories are 0s.

This target formulation enables the network to be trained directly to align the classification and localization tasks. It learns to penalize predictions that are highly confident in terms of classification but poorly localized. This way, best-fitting boxes survive the NMS process [30], and the confidence output is unified and high-quality.

3.2.4 Loss Function

The total loss is a weighted sum of the Bounding Box Regression Loss, Classification, and Distribution Focal Loss (DFL):

$$\begin{aligned}
\mathcal{L}_{total} &= \lambda_{box} \cdot \mathcal{L}_{box} + \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{dfl} \cdot \mathcal{L}_{dfl} \\
&= \lambda_{box} \left(\frac{\sum_{i \in fg} (1 - CIoU_i) \cdot weight_i}{\sum_i target_scores_i} \right) \\
&\quad + \lambda_{cls} \left(\frac{1}{\sum_i target_scores_i} \sum_i BCE(pred_scores_i, target_scores_i) \right)
\end{aligned}$$

$$+ \lambda_{dfl} \left(\frac{\sum_{i \in fg} \left(w_{l,i} \cdot CE(pred_i, t_{l,i}) + w_{r,i} \cdot CE(pred_i, t_{r,i}) \right) \cdot weight_i}{\sum_i target\_scores_i} \right), \quad (3.5)$$

where λ_{box}, λ_{cls}, and λ_{dfl} are hyperparameters that balance the contributions of the three loss components. The index i corresponds to an individual anchor point. BCE denotes the Binary Cross-Entropy (BCE) loss used for multi-label classification, while CE denotes the Cross-Entropy (CE) loss used within the DFL to supervise the predicted boundary distribution. The variable fg represents the foreground mask, i.e., the set of anchor points assigned to ground-truth objects.

The $target\_scores_i$ are defined in Equation 3.4. The variables $t_{l,i}$ and $t_{r,i}$ denote the discrete left and right integer bounds $(\lfloor target_i \rfloor, \lfloor target_i \rfloor + 1)$ of the target box coordinate for anchor point i. Finally, $w_{l,i}$ and $w_{r,i}$ are interpolation weights defined as $w_{l,i} = t_{r,i} - target_i$ and $w_{r,i} = 1 - w_{l,i}$.

The interpretation of the individual losses is as follows: Classification Loss independently evaluates the accuracy of the predicted object categories, using Binary Cross-Entropy. Bounding Box Regression Loss penalizes inaccuracies in the predicted bounding box coordinates. It uses the Complete Intersection over Union (CIoU), which considers not only the overlap between the predicted and ground truth boxes but also the distance between their centers and shape accuracy. Distribution Focal Loss allows the model to predict a probability distribution for box edges. The model achieves highly precise localization even when object boundaries are ambiguous or partially occluded by supervising the network to focus on discrete integer bounds (t_l, t_r) using Cross-Entropy.

3.3 Dataset Limitations and Synthetic Dataset Creation

Computer vision models used for construction equipment detection face significant limitations due to the scarcity of annotated training data and the high cost of manual annotation [12]. This creates a gap between the training data and real-world site conditions, reducing the model's adaptability to varying lighting conditions, environmental factors, and specific equipment models.

Recent advancements emphasize a data-centric framework that automatically generates site-specific synthetic datasets, overcoming this limitation without relying on extensive manual annotation [12]. This involves creating new training images by combining different elements. Firstly, equipment is separated from its original background using zero-shot instance segmentation, such as the Segment Anything Model 2 (SAM2) [31]. Secondly, depth estimation models, such as Depth Anything 2 (DA2) [32], are used to generate background depth maps, enabling the equipment to be placed within the image at the correct scale [12].

3.4 Vision-Language Models for Classification Refinement

Vision-Language Models introduce a unique combination of visual comprehension, human-like world knowledge, and robust reasoning abilities into the detection pipeline. More advanced VLMs, such as GPT-4o, LLaVA, and Qwen2-VL, have demonstrated significantly enhanced capabilities for image and text understanding [33]. A primary advantage of these pre-trained VLMs is their strong few-shot learning capabilities and robustness, which is highly beneficial in construction environments where annotated data is typically scarce.

Advanced generative VLMs offer an open-vocabulary, descriptive paradigm, not just an assignment of rigid class numbers. Their pre-training provides a broad understanding of the world, so modern VLMs can analyze an isolated image crop of an object and generate an accurate, concise textual description of its physical state. Crucially, this deep semantic understanding enables an automated system to evaluate a specific tool's level of danger based on its visual condition, even if the system has not been explicitly trained on a rigid class for that exact brand or variant.

One recent publication that applies VLMs for construction site monitoring is presented by Chen and Yin [33]. As fully fine-tuning such massive models to recognize specific construction hazards would demand considerable computational resources and memory, the authors employ the Low-Rank Adaptation (LoRA) technique [34]. Low-Rank Adaptation is utilized not only to mitigate this memory overhead but also to prevent catastrophic forgetting and maintain inference efficiency during task-specific adaptation. This technique 'freezes' the pre-trained 'brain' (represented as W_0), so the model doesn't forget the general knowledge. Then, the only two tiny new matrices (A and B) must be trained and added together: $W = W_0 + AB$. These matrices are the tiny updates that teach the VLM what a specific site looks like [33].

To properly evaluate the model's performance in a high-stakes environment, Chen and Yin [33] utilized the $\boldsymbol{F_2}$ **Score** to prioritize safety over convenience in construction hazard detection. It is derived from the generalized F_β formula:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}. \tag{3.6}$$

By setting $\beta = 2$, this formula assigns a much higher weight to Recall (finding all hazards) over Precision (avoiding false alarms), ensuring that their model is evaluated based on its ability to detect potential hazards as accurately as possible [33].

4 Methodology

4.1 System Overview

This thesis aims to construct a two-stage framework combining fast detection and Vision–Language Reasoning, in order to recognize different tools and equipment on the construction sites.

In the first stage, we apply a single-class YOLOv12 architecture to identify objects within the 'tool' category, rather than utilizing a multi-class approach. This design choice provides essential scalability, as construction environments frequently introduce new tool variants. The system remains robust and also avoids the computational overhead of frequent multi-class retraining.

Although YOLOv12 is excellent at real-time spatial localization, it lacks the semantic reasoning necessary to evaluate the functional danger level of a detected object. Therefore, the second stage of the proposed pipeline integrates a cloud-based Vision-Language Model (VLM). By passing cropped bounding boxes to the VLM, it generates an open-vocabulary description. This way, the autonomous robot can perform a contextual risk assessment.

4.2 Dataset Preparation and Augmentation

For training YOLOv12, a synthetic tools dataset was created. Its main characteristic is realistic, reflecting real-world floor conditions on a construction site. Each image shows one tool lying on the construction site floor (concrete, dirt, sand, asphalt, tiles). The robot's camera is mounted to face downward, so the dataset must reflect this reality. Therefore, background images resemble a robot's perspective, avoiding a human or drone's point of view. Additionally, the tools are realistically placed and sized.

Firstly, 15 tool classes were chosen from the 'Diverse Tools Image Dataset for Machine Learning' [35]. The selection was based on tools commonly used in construction environments, many of which are hazardous, but not all. The classes include cone, drill, float, grinder, hammer, knife, nail gun, paintbrush, pry bar, saw, spirit level, tackle, measuring tape, trowel, and wrench. For each tool type, 8 images were selected, yielding a total of 120 tool and equipment images. Secondly, the background was

removed using Remove.bg [36] and Canva [37]. Thirdly, all the images were manually labeled in Roboflow [38].

For the augmentation part, a Python script was created with the following pipeline: foreground preparation, random object placement into backgrounds, and photorealistic augmentation. This pipeline is shown in Figure 4.1.

Each tool type is processed separately. The images are loaded with an alpha channel transparency (RGBA format) to create 'invisible' pixels for a background. The bounding box annotations are extracted from JSON metadata files that contain the image dimensions and the normalized tool.

Tools are randomly rotated up to $\pm 20^{\circ}$ with a 100% probability to simulate realistic handling variations in workshop scenarios. Tool sizes are scaled relative to the background dimensions using tool-specific thresholds, such as 20-30% for hammers, 30-40% for drills, and 5-15% for knives. These thresholds are based on standard tool sizes and were manually adjusted for certain images because they did not look realistic in the dataset.

These spatial proportions are further refined by ensuring that the bounding box occupies between 3% and 15% of the total image area, that at least 75% of the object is below the midline, and that the entire tool remains within the image boundaries. The upper 25% of the background is always left empty because the tool would otherwise appear to be flying or too far away, from the robot's view, which would not represent a realistic hazard.

There are 7 background images where tools are placed to ensure diversity. During random placement, alpha blending is applied to produce smooth and realistic edges.

For each tool and for each background, 20 images are generated. Sometimes, fewer images are produced because the number of attempts to paste the tool into the background is limited to avoid loops. Therefore, the overall size of the dataset is not exactly 15 categories $\times$ 8 tools/category $\times$ 7 backgrounds $\times$ 20 images/background $=$ 16800, but slightly smaller, around 13,500 images, which is still sufficient to provide a qualitative and realistic dataset.

The next step is providing augmentation. It prevents the model from overfitting to the inherent perfection of synthetic data. The script introduces stochastic variations in brightness ($\pm$8%) and contrast ($\pm$15%), along with random shadows and Gaussian blur. These transformations, combined with horizontal flipping to ensure geometric invariance, force the network to prioritize robust structural features over transient pixel values.

Finally, the generated images are automatically partitioned into training, validation, and test sets using an 80/10/10 split. Each image has a corresponding YOLO-format annotation $(class\_id, x\_center\_norm, y\_center\_norm, width\_norm, height\_norm)$. Since we have only one class, *class_id* is always equal to zero.

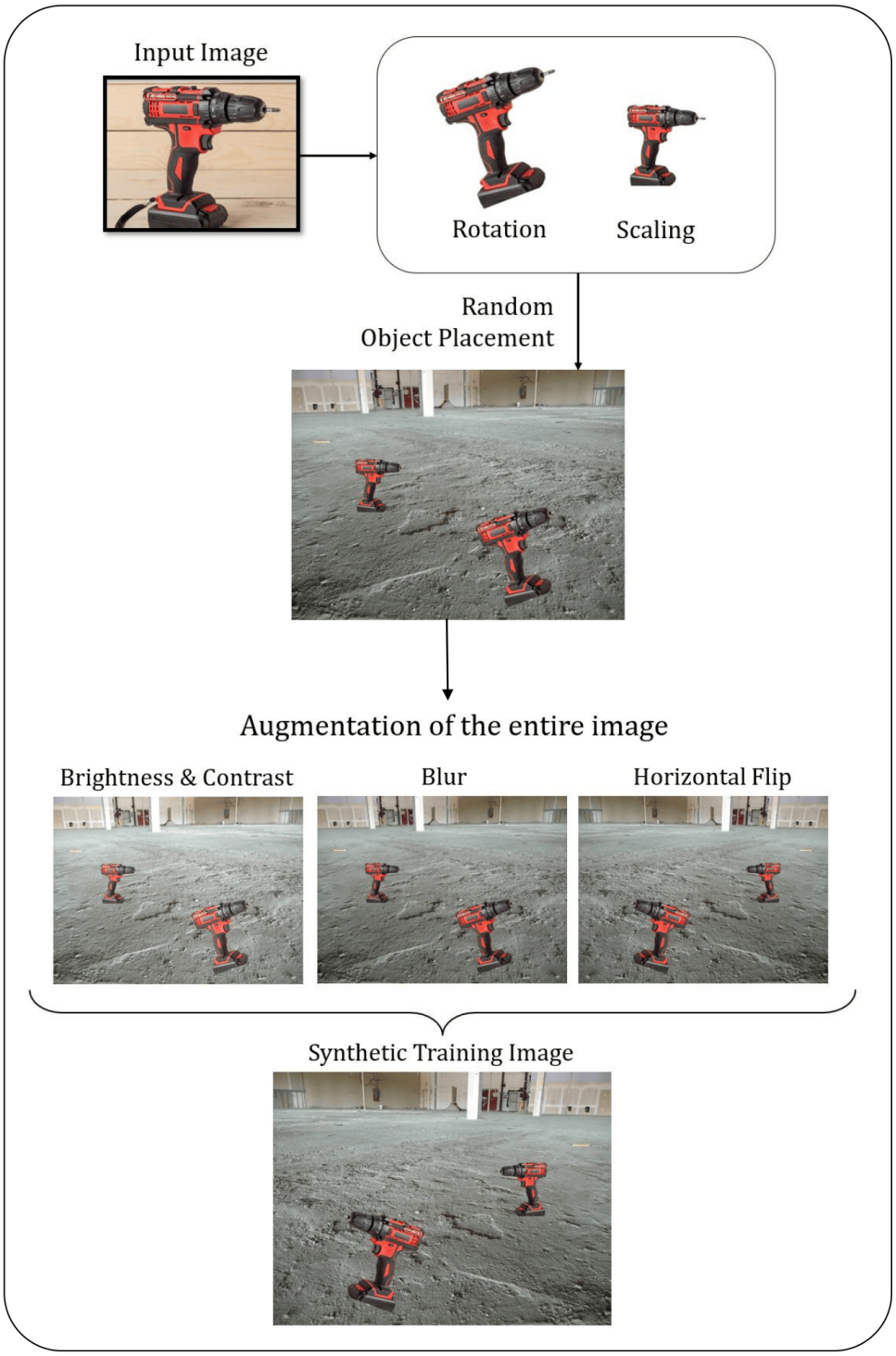


Figure 4.1: Augmentation workflow.

4.3 YOLOv12 Implementation and Single-Class Adaptation

Since we use a single-class YOLO model, the initial idea was to remove the classification term from the loss function entirely. This, unfortunately, led to a low confidence score regardless of the number of epochs. The reason was that the confidence score combines classification and objectness using IoU-Aware Target Assignment. High scores are reserved for cases where both the predicted classification and the bounding box's spatial localization are accurate, as discussed in 'Confidence Scoring in YOLOv12'. Therefore, while the YOLOv12 standard architecture was maintained, all ground truth labels were assigned to a single universal tool class.

Practically, this was implemented by defining the dataset configuration file (tool.yaml) with the number of classes strictly set to one (nc: 1), and ensuring that the class identifier in every corresponding bounding box annotation file (.txt) was formatted as 0. The dataset configuration is presented in Listing 4.1.

```
path: /content/datasets
train: images/train # train images
val: images/val # validate images
test: images/test # test images

nc: 1

# Classes
names:
  0: tool
```

Listing 4.1: Custom dataset configuration (`tool.yaml`).

4.4 Vision-Language Reasoning Integration

From the previous stage, YOLOv12 identifies a tool and outputs the bounding box coordinates (x_1, y_1, x_2, y_2). These coordinates are used to mathematically crop the tool from the original camera image. By isolating the tool this way, the cloud-based VLM only receives an image of the target object, avoiding distractions from the surrounding background.

To transmit this cropped image to the cloud, it is first converted to a Base64 string and packaged into a JSON payload along with a designed instruction prompt. It does not merely request a simple textual description of the tool, but instructs the VLM to perform three tasks simultaneously: identifying objects, assessing hazards,

and generating an avoidance recommendation for the robot. This ensures the model returns its analysis in a structured JSON format that the navigation system can interpret directly.

The structured instruction used for the VLM is shown in Listing 4.2.

```
You are an expert in recognizing construction tools and equipment on
construction sites.

The robot is a Husky mobile manipulator with wheels that navigates autono-
mously on the ground. Your task is to analyze a cropped
image that should contain a single construction tool or
piece of equipment lying on the ground.

Goals:
1. Identify the object.
2. Assess the hazard for robot navigation.
3. Provide an avoidance recommendation.

Return valid JSON only with the following fields:
- label (string)
- confidence ("low"|"medium"|"high")
- hazard ("yes"|"no"|"uncertain")
- hazard_type (list of strings)
- severity ("low"|"medium"|"high")
- avoidance (string, one sentence instruction for the robot)
- notes (string, optional; mention occlusion/blur or alternative labels)

Safety rule: If uncertain, mark hazard="uncertain" and choose conserva-
tive avoidance.
```

Listing 4.2: Unified prompt used for the Vision-Language Model hazard assessment.

The entire data package is then sent to the cloud-based VLM (e.g., OpenAI [39]) via a standard web API call. By receiving the data and the prompt, the VLM analyses the tool's physical features (e.g., sharp edges and heavy objects), evaluates its hazard level and generates avoidance recommendations.

An example of a possible response generated by the VLM is shown in Listing 4.3.

```
{
  "label": "utility knife",
  "confidence": "high",
```

```
  "hazard": "yes",
  "hazard_type": ["sharp"],
  "severity": "high",
  "avoidance": "keep 1.0 m clearance and navigate around the object",
  "notes": "exposed blade visible"
}
```

Listing 4.3: Example VLM response for a detected hazardous tool.

Based on the VLM's risk assessment, the robot's navigation system can dynamically adjust its driving path to proactively avoid a potentially dangerous tool and ensure safe operations on the construction site floor.

5 Results and Discussion

This chapter details the experimental setup, training procedures, and the computational environment used to develop and evaluate the construction site hazard assessment model.

5.1 Dataset Configuration

The model was trained using a custom dataset synthesized from 120 tool images from 15 categories and 7 background images, as described in Section 4.2. The final dataset contains approximately 13500 images, which were partitioned into training, validation, and test sets with 80%, 10%, and 10% splits, respectively.

To address the scarcity of robotic perspective data, background images were sourced from public repositories [40] and supplemented with manually captured images to ensure environmental diversity.

5.2 Experimental Setup

The training and evaluation were conducted in a high-performance cloud computing environment provided by Kaggle [41]. The experiments were executed on Kaggle's GPU-enabled notebooks, using an NVIDIA Tesla T4 GPU with 30 GB of system RAM and 20 GB of available disk space. All model training and evaluation were performed using YOLOv12 as discussed in Section 4.3. The YOLOv12 model was trained using SGD with an initial learning rate that was gradually adjusted during optimization. To improve model generalization and prevent overfitting, various data augmentation techniques were incorporated into the training process. The model was trained for 100 epochs with an input image resolution of 640 × 640 pixels. Additionally, a batch size of 16 was selected to ensure efficient training, as using a batch size of 1 significantly increased training time.

5.3 Experimental Results and Analysis

5.3.1 Comparison of Augmentation Strategies

The study was conducted to evaluate the impact of various data augmentation strategies and justify the selected parameters. Dataset augmentations were applied during dataset creation using the optimized augmentation pipeline. Additionally, YOLOv12 applies on-the-fly internal augmentations during training (e.g., mosaic, HSV color shifts, scaling), intentionally exposing the model to high variance to boost accuracy.

For Runs 1–3, YOLOv12's default augmentations were enabled during training, while for Run 4, all augmentations were disabled to provide a strict control baseline. Four distinct training runs were executed:

- **Run 1 (Baseline Augmentation):** Utilized the optimized augmentation pipeline, including `RandomBrightnessContrast` (brightness_limit=0.08, contrast_limit=0.15, $p = 0.4$), `GaussianBlur` (blur_limit=(3, 5), $p = 0.3$), and `HorizontalFlip` ($p = 0.5$).
- **Run 2 (Extreme Contrast):** Applied an aggressive contrast limit of 0.8 to test the model's robustness under severe lighting variations.
- **Run 3 (Extreme Blur):** Increased the Gaussian blur kernel size to (11, 15) to simulate severe camera defocusing or heavy motion blur.
- **Run 4 (No Augmentation):** Served as a strict control group. Aside from the baseline rotation and scaling applied during the initial synthetic dataset generation, no further augmentations were introduced. For this control run, these internal processes were explicitly disabled by setting `augment=False` in the training configuration, ensuring a pure, unaugmented baseline.

Figure 5.1 illustrates the convergence of the overall model across the four runs by comparing the total validation loss. Furthermore, Figure 5.2 provides a detailed comparison of key evaluation metrics: Precision, Recall, mAP@50, and mAP@50-95, recorded over the 100 training epochs.

Overall, the training behavior of all four runs is very similar. The total validation loss decreases quickly during the first few epochs and then stabilizes. After about 50 epochs, the curves begin to level off, indicating that the model has largely converged and that further training yields only small improvements.

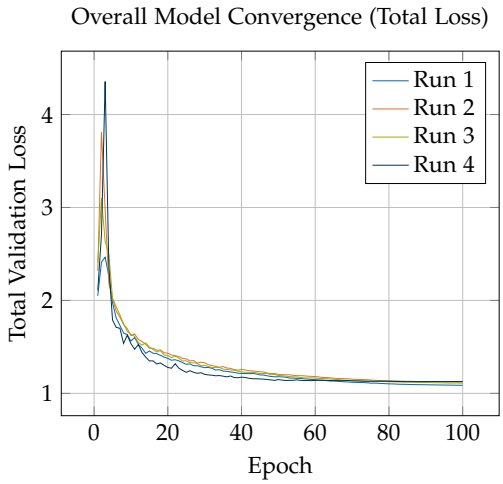
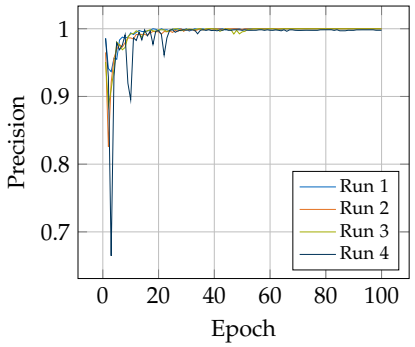
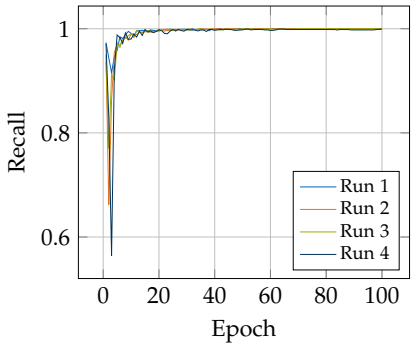


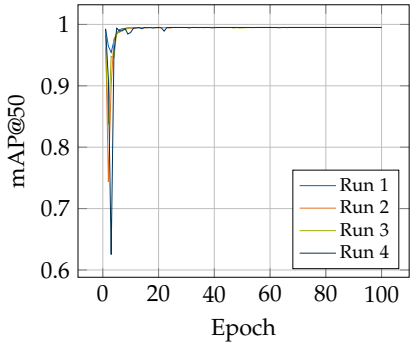
Figure 5.1: Comparison of total validation loss across four runs. The total loss is the sum of Box, Class, and DFL losses.



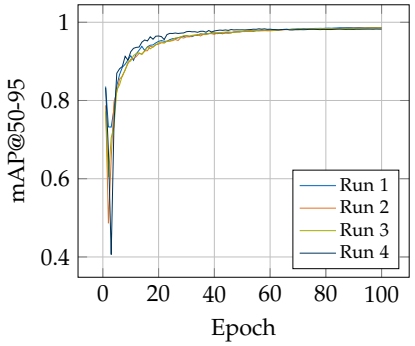
(a) Precision comparison



(b) Recall comparison



(c) mAP@50 comparison



(d) mAP@50–95 comparison

Figure 5.2: Comparison of evaluation metrics across four training runs. Precision, recall, mAP@50, and mAP@50–95 are plotted over training epochs.

A comparison of the evaluation metrics further supports this observation. The precision and recall values increase rapidly over the first few epochs and stabilize near 1.0. This means the model quickly learns to correctly detect and classify the tools. The mAP@50 metric also reaches very high values early in training, whereas the more stringent mAP@50–95 metric improves more gradually because it requires more accurate bounding box localization and stabilizes at a high level by the end of training.

When comparing the four experimental runs, only minor variations in the augmentation strategies are observed. The baseline augmentation configuration (Run 1) shows slightly smoother convergence, whereas the extreme augmentation settings in Runs 2 and 3 introduce slightly higher variability in the early training stages. Nevertheless, these differences diminish as training progresses.

Run 4 shows a significantly larger loss spike during the first few epochs compared to the other configurations. The absence of augmentation reduces the variability of the training data, which can initially make the optimization process less stable when combined with the relatively high learning rate. Despite this early instability, Run 4 eventually converges to a similar loss level as the other runs. This suggests that the model can still learn effective feature representations even without augmentation.

Another observation across all runs is the noticeable spike in loss during the very first training epochs. This behavior is likely caused by the relatively high initial learning rate of 0.01 and the absence of a learning rate warm-up phase. At the beginning of training, the optimizer performs large weight updates, temporarily increasing the loss before the model stabilizes and begins to converge.

5.3.2 Learning Rate Comparison

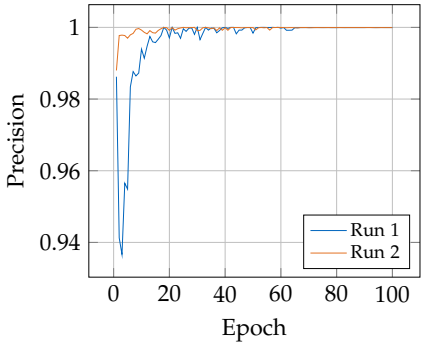
To investigate whether a lower learning rate would lead to more stable training during the early epochs, an additional experiment was conducted. In this comparison, Run 1 used a learning rate of 0.01, while Run 2 used a smaller learning rate of 0.001. The results are shown in 5.3, including Precision, Recall, mAP@50, and mAP@50–95.

Overall, the final performance values achieved by both runs are very similar. Precision and recall quickly increase and stabilize at around 1.0 in both runs, indicating that the model learns to correctly detect and classify the tools early in training. Both runs achieve very high mAP@50 values after only a few epochs, reaching approximately 0.99, and similar final mAP@50-95 values close to 0.98.

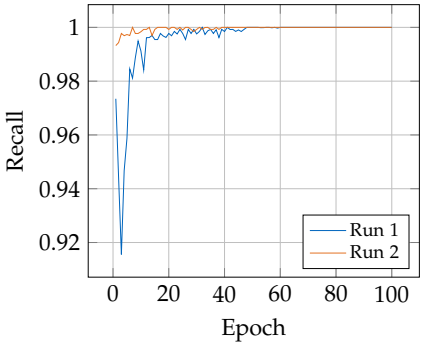
Run 1 demonstrates stronger spikes during the first few epochs in all metrics. In contrast, Run 2, which uses a smaller learning rate of 0.001, shows a smoother and more stable increase. This behavior indicates that a higher learning rate leads to stronger weight updates during the initial training stages, potentially causing temporary instability in the optimization process. A smaller learning rate enables the model to

learn more gradually, resulting in more stable training curves.

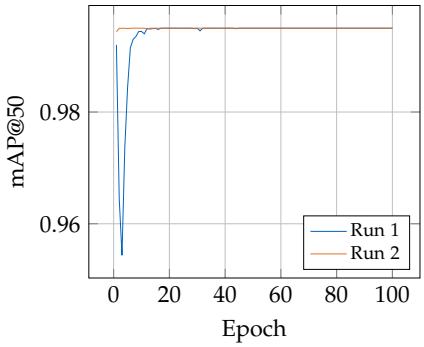
Due to time constraints, the remaining experiments in this study were conducted using the original learning rate of 0.01. However, using a smaller learning rate could improve training stability in future work.



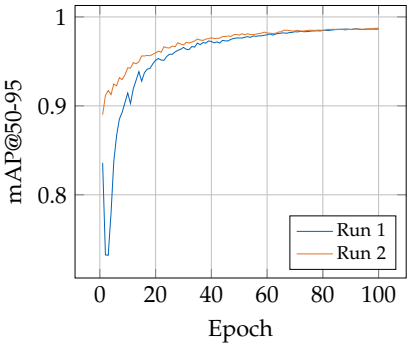
(a) Precision comparison



(b) Recall comparison



(c) mAP@50 comparison



(d) mAP@50–95 comparison

Figure 5.3: Comparison of evaluation metrics for two training runs using different learning rates.

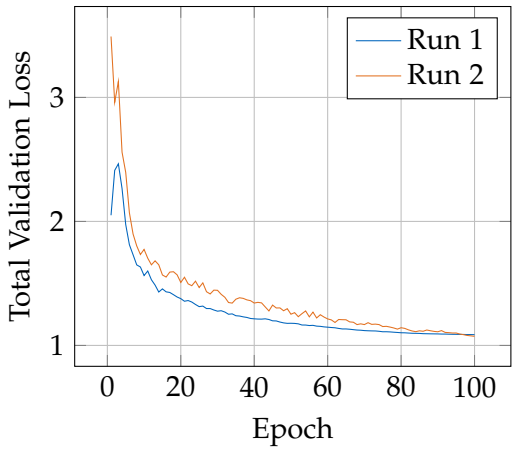
5.3.3 Dataset Size Analysis

Another experiment was conducted to evaluate the influence of dataset size on model performance. Two synthesized datasets were compared: a smaller dataset containing approximately 1400 images, and a larger dataset, consisting of approximately 13500 images, used in all other main experiments.

Figure 5.4 compares the total validation loss during training for both datasets. Run 1 (full dataset) exhibits more stable training behavior, whereas Run 2 (small dataset)

shows frequent fluctuations in the validation metrics before converging. Although the final mAP@50 results are identical, the mAP@50-95 is slightly higher for the smaller dataset. This means the model can achieve peak performance because the detection task is relatively simple.

These findings confirm that increasing the dataset size improves the model's ability to generalize by exposing it to a greater variety of object placements, scales, and background conditions. Therefore, the larger dataset provides a more reliable training setup for the hazard detection task.



(a) Validation loss comparison.

Dataset	Images	mAP@50	mAP@50-95
Small dataset	1400	0.995	0.995
Full dataset	13500	0.995	0.987

(b) Detection performance.

Figure 5.4: Comparison of validation loss and detection performance for two dataset sizes.

5.3.4 Influence of Contrast Augmentation

Lightness and contrast conditions can vary significantly on construction sites, depending on whether artificial lighting is present and on outdoor conditions. Several contrast augmentation levels were tested to determine the optimal contrast percentage for the

model. The results are presented in Table 5.1, which summarizes the detection performance obtained with contrast limits of 15%, 30%, 50%, and 80% with a probability of $p = 0.4$.

Table 5.1: Performance Comparison for Different Contrast Augmentation.

Contrast	mAP@50	mAP@50-95
15%	0.995	0.987
30%	0.995	0.986
50%	0.995	0.984
80%	0.995	0.986

The results demonstrate that all configurations achieve identical mAP@50 values of 0.995. However, minor differences can be observed in the stricter mAP@50–95 metric, where a baseline contrast value of 15% achieves the best performance (0.987). Therefore, while the model is highly robust to variations in image contrast, increasing contrast beyond 15% offers no gain in detection and may disrupt image features.

5.3.5 Influence of Blur Augmentation

Blur augmentation was evaluated to simulate situations such as camera defocus or motion blur that may occur in a robotic system. Three Gaussian blur configurations were compared: the baseline setting of (3, 5), a moderate blur of (7, 9), and a stronger blur of (11, 15) with a probability of $p = 0.3$.

The results presented in Table 5.2 show that all setups have identical mAP@50 values of 0.995. The baseline blur configuration (3,5) achieves the highest mAP@50-95 score of 0.987, while stronger blur settings slightly reduce performance to approximately 0.985. This behavior can be explained by the strong blurring effect on image features, such as object edges, which are necessary for accurately locating bounding boxes. Although the model can still detect objects correctly, the predicted bounding boxes become slightly less precise.

The baseline configuration (3, 5) provided the best balance between robustness and

Table 5.2: Performance Comparison for Different Blur Augmentation.

Gaussian Blur	mAP@50	mAP@50-95
(3, 5)	0.995	0.987
(7, 9)	0.995	0.985
(11, 15)	0.995	0.985

detection accuracy and was kept for the final training.

5.3.6 Final Analysis

Quantitative Evaluation

One additional comparison was conducted to assess whether the model overfitted. For this purpose, the standard evaluation metrics, including Total Loss, mAP@50, mAP@50-95, Precision, and Recall, were compared between the test and validation datasets. The results are presented in Table 5.3.

Table 5.3: Performance Comparison Between Validation and Test Sets.

Dataset	**Total Loss**	**mAP@50**	**mAP@50-95**	**Precision**	**Recall**
Test	1.0970	0.995	0.9865	0.9999	1.0
Validation	1.0867	0.995	0.9867	1.0	1.0

The performance metrics for both datasets are very similar. The mAP@50 values are identical, and both Recall values reach 1.0. Precision is also nearly identical for the two datasets, meaning that almost all predicted detections correspond to correct objects. The Total Loss is slightly higher for the test set, while the mAP@50-95 is marginally lower than for the validation set.

These small differences indicate that the model generalizes well between the validation and test datasets and shows no signs of significant overfitting. The nearly identical performance across both datasets suggests that the training process and the applied augmentation techniques provided sufficient variability for the model to learn robust features.

Evaluation on Real-World Images

To further assess the model's behavior, several real-world images that were not included in the dataset were analyzed. These images contained tools such as a drill, hammer, saw, spirit level, and a wrench, as well as some unknown tools not present in the training data. In addition, other objects appeared in the images, including a trash basket, a box containing various small tools, a part of a wooden plank, and a warning tape located on the floor.

The model correctly identified the primary tool in 14 out of 43 images (11 of which contained unknown tools not present in the training dataset). The most problematic cases occurred when the tool was lying on the warning tape or was upside down. Examples of both a correctly detected tool and a missed detection are illustrated in

Figure 5.5. The result suggests that the diversity of the training images could be improved by including additional augmentations, such as vertical transformations, and by introducing more variations in floor textures and backgrounds.

All detected outputs showed the same false positive on a measuring tape. This is probably because certain shapes or edges were similar to visual features that the model associates with tools.

Although unknown tools were rarely detected, the box containing multiple small tools was frequently recognized as a tool. This behavior may indicate that the model responds to the presence of tool-like shapes within the box.

Overall, these observations highlight that while the model performs well on quantitative metrics, its robustness in real-world environments can be improved by increasing dataset diversity and including a wider range of object orientations and background conditions.

Figure 5.5: Example detection results on real-world images.

5.3.7 Runtime Performance

To evaluate the runtime performance of the detection stage, processing time was measured across 120 images from the created dataset on a single NVIDIA T4 GPU. The runtime was measured using the model's prediction function, which includes image preprocessing, model inference, and bounding box generation. The average processing time was 0.075 seconds per image. This result shows that the detection model can operate at near real-time speeds under the tested hardware conditions.

5.4 Vision-Language Reasoning Evaluation

To evaluate the performance of the Vision-Language Model (VLM), the bounding boxes of some tools, generated by the YOLOv12 detection stage, were cropped and sent to

a cloud-based model via the OpenAI API [39]. For each detected object, the system sent the string of a cropped image together with the structured prompt described in Section 4.4. The GPT-based VLM analyzed the image and returned a structured JSON response containing the identified object, a hazard assessment, and an avoidance recommendation for the robot.

The findings show that the VLM can perform semantic reasoning about construction tools, equipment, and their potential hazards. In the most tested cases, the model successfully identified common tools and evaluated their associated risks based on visible characteristics, such as sharp edges or metallic components. However, in some examples, a construction float lying on the floor was incorrectly interpreted by the VLM as a knife-like object or as a trowel, which is visually similar. The model still detected a potentially sharp metallic edge and labeled the tool as dangerous, thereby preserving the system's safety objective even though the exact tool category was misidentified.

Still, for all other categories, the tool was generally correctly classified and described. An example response of a system evaluation is shown in Listing 5.1. In this example, the model identifies a hand trowel and classifies it as a potential hazard due to its sharp metal blade and the possibility of obstructing the robot's path. The model also generates an avoidance recommendation suggesting that the robot maintain a clearance distance of 0.5 meters.

```
{
  "label": "hand trowel",
  "confidence": "high",
  "hazard": "yes",
  "hazard_type": ["sharp object", "trip hazard", "mobility obstruction"],
  "severity": "medium",
  "avoidance": "Navigate around the hand trowel with a minimum clearance
   of 0.5 meters.",
  "notes": "Metal blade may pose a puncture risk to robot wheels or
   humans."
}
```

Listing 5.1: Example VLM response generated through the OpenAI API.

The structured JSON output enables the robot navigation system to analyse the model's reasoning directly, by programmatically extracting fields such as *hazard*, *severity*, and *avoidance*.

Overall, integrating the GPT-based VLM extends traditional object detection by enabling a deeper understanding of the tool or equipment. This allows the robot not only to detect tools but also to evaluate their associated risks and adapt its navigation strategy accordingly.

5.4.1 Qualitative Comparison Between Cropped Detections and Raw Images

An additional experiment was conducted to compare the VLM's outputs when using cropped versus full images. For this experiment, all present in the dataset tools with a random background were analyzed twice: first with the full image and then with a cropped version, focusing on the object. The results of the classification comparison are summarized in Table 5.4.

Tool	Cropped Images (Correct / 8)	Raw Images (Correct / 8)
Cone	8 / 8	8 / 8
Drill	8 / 8	8 / 8
Float	2 / 8	2 / 8
Grinder	8 / 8	8 / 8
Hammer	8 / 8	5 / 8
Knife	8 / 8	7 / 8
Nail gun	8 / 8	8 / 8
Paintbrush	8 / 8	8 / 8
Pry bar	7 / 8	6 / 8
Saw	8 / 8	8 / 8
Spirit Level	8 / 8	8 / 8
Tackles	7 / 8	6 / 8
Measuring Tape	7 / 8	7 / 8
Trowel	8 / 8	7 / 8
Wrench	8 / 8	7 / 8

Table 5.4: Performance Comparison of VLM Classification Using Cropped and Raw Images.

For most medium- and large-sized tools (e.g., drill, hammer, cone, grinder, spirit level, nail gun, saw, trowel), the model produced nearly identical outputs for both cropped and full images. This suggests that the model can reliably recognize these objects regardless of whether the surrounding context is present.

For smaller tools (e.g., paintbrush, float, measuring tape), the outputs were generally similar for cropped and full images, although several inconsistencies occurred. One notable case involved the concrete float: only 2 out of 8 images were correctly classified, while in the remaining cases the model predicted a trowel, likely due to the strong visual similarity between these two masonry tools.

Additional misclassifications were also observed. For instance, a knife was occasionally recognized as a screwdriver when the full image was used, while a set of tackles was misidentified as a caulking gun. Another example involves a pry bar, where the

object label was incorrect despite the hazard being correctly detected, as shown in Table 5.5 and Figure 5.6. These results suggest that the surrounding context in the full image can influence the model's object recognition, even though the overall hazard assessment often remains consistent.

Finally, the paintbrush was sometimes classified as a hazard in both cropped and full images, even though the object itself was correctly identified. However, this issue can be resolved by modifying the prompt and providing additional contextual information about the robot system. For example, specifying that the robot is equipped with wheels and can safely move over small or non-sharp objects. With this additional context, the hazard classification was corrected while the object identification remained accurate.

In summary, the majority of the model outputs were correct. However, the results indicate that the recognition accuracy decreases for very small or narrow tools, where object features are less prominent and may be more difficult for the model to distinguish.

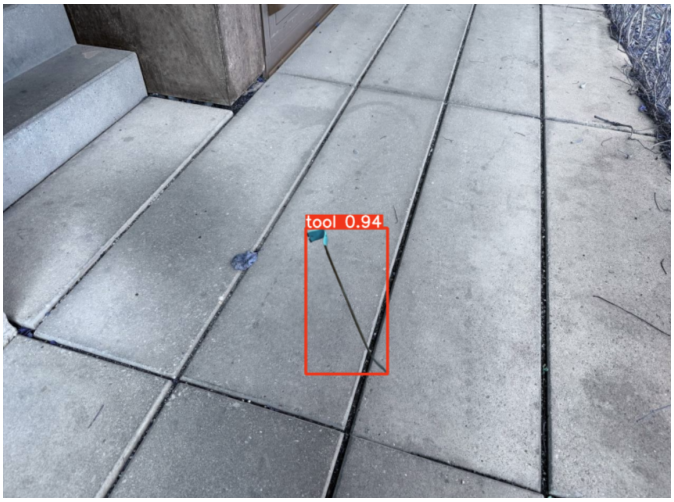


Figure 5.6: Example detection of a pry bar.

5.4.2 Real-World Image Evaluation

The same 43 real-world images analyzed in Subsection 5.3.6 were also used to evaluate the performance of cropped and raw image recognition using the VLM.

For the cropped images, the VLM could only analyze the 14 tools that had previously been detected by the YOLO model. In these cases, the VLM performed well, correctly classifying the objects without noticeable errors.

In raw images, the model sometimes failed to correctly classify the primary tool when using the original prompt. Initially, it was designed to identify only a single tool, which

caused difficulties when multiple objects were present in the background. In such cases, the model occasionally focused on background elements instead of the intended object. When the prompt was modified to explicitly request identification of multiple tools in the image, the model generally performed well, correctly identifying them. However, images contained in a toolbox were sometimes omitted from the generated description. Additionally, the model sometimes classifies a measuring tape as a tool, similar to earlier observations of the object detection model associating certain environmental structures with tool-like features.

To sum up, these results suggest that the VLM performs more reliably when provided with cropped object detections. When raw images are used, the model sometimes fails to identify the primary tool because additional background elements distract the model from the object of interest.

Field	**Cropped Image**	**Full Image**	**Agreement**
Label	Crowbar	Utility marking flag	No
Confidence	High	High	Yes
Hazard	Yes	Yes	Yes
Hazard Type	Tripping, obstruction, puncture	Tripping, obstruction, entanglement	Partial
Severity	Medium	Medium	Yes
Avoidance	Navigate around the object, maintaining at least 0.5 meters distance.	Navigate around the flag at a safe distance to avoid contact with the pole.	Partial
Notes	The tool is lying at an angle and may partially obstruct the path, creating a tripping and puncture hazard.	The thin metal flag pole may entangle wheels or trip sensors; flag is clearly visible and not occluded.	Partial

Table 5.5: Comparison of Hazard Annotations.

6 Conclusion and Future Work

6.1 Conclusion

In construction sites, real-time object detection and semantic understanding are crucial for robotic systems to operate safely and avoid incidents. However, conventional object detectors mainly rely on classification and do not provide a deeper contextual understanding of the detected equipment. This thesis presented a two-stage framework that combines object detection with vision–language reasoning for construction site tools and equipment, aiming to improve both detection accuracy and semantic interpretation.

First, a synthetic dataset was created by combining cropped tool images and pasting them onto different background scenes to simulate realistic environments. Second, a YOLOv12 model with a single-class detection configuration was trained on this dataset to detect tools and equipment. Finally, the detected objects were cropped and analyzed using a Vision-Language Model, which generated textual descriptions and assessed potential hazards associated with the tools.

Several experiments were conducted to evaluate the proposed framework, including comparisons of different augmentation strategies and hyperparameter configurations, overfitting analysis, real-world objects detection, and evaluation of VLM outputs. Overall, the experiments demonstrate that the proposed training pipeline achieves consistently high detection performance across all configurations. The final model reached an mAP@50 score of approximately 99.5% and an mAP@50-95 score of around 98.7%. Precision and recall both reached 100% on the evaluation dataset, indicating that the model detected all annotated tools without false positives. The comparison between validation and test datasets showed nearly identical results, confirming that the model generalizes well and does not exhibit significant overfitting.

In addition, qualitative experiments were conducted on real-world images to evaluate the framework in practical scenarios. The detection model correctly identified the primary tool in 14 out of 43 images from previously unseen environments, and the VLM generated meaningful hazard descriptions for most detected objects. While the outputs were generally consistent across cropped and full images, some inconsistencies were observed with smaller or visually similar tools, suggesting that object size and visual similarity can affect recognition accuracy.

In summary, the results demonstrate that integrating object detection with vision-

language reasoning effectively enhances a robot's ability to detect and analyze tools and equipment in construction environments.

6.2 Future Work

For future work, we propose several improvements for the two-stage framework. The dataset could be extended to include additional objects commonly found on construction sites, though not necessarily hazardous. For instance, personal equipment such as mobile phones, laptops, or cameras may lie on the floor and should be detected to avoid damage during robot navigation. Increasing the number and diversity of tool images would also improve the detection model's robustness.

Another possible improvement would be to automate the selection of suitable background images for the synthetic dataset generation. Currently, background images are manually selected to match the robot's viewing perspective. An automated pipeline could analyze candidate images and classify them based on their viewpoint and scene structure. VLMs could be used to identify images that align with the robot's perspective. The same approach could be used to choose tools from an open-source dataset. Moreover, creating artificial images with the same tools but different backgrounds could help evaluate the model's ability to generalize, since the current test set follows a distribution similar to the training data.

Additionally, the dataset could include scenes with multiple tools on the floor, creating more crowded, realistic construction site environments. Training the model on such scenarios could improve its ability to detect objects under more complex conditions. Furthermore, future research could optimize the VLM on construction-specific datasets to improve the accuracy and consistency of tool identification and hazard reasoning.

In the current workflow, data augmentation is applied during dataset generation and during model training. Future work could investigate applying augmentation only during training, which may further increase data variability across training epochs.

Finally, the proposed framework could be integrated into a real-time robotic system. Such a system would process continuous video streams, enabling the robot to detect and track tools in real time. This would provide further insight into the practical applicability and performance of the approach in dynamic real-world construction environments.

Overall, these future directions highlight the potential of the proposed framework to evolve into a more robust and intelligent perception system, advancing open-world understanding of construction tools and equipment and supporting safer autonomous operation in complex construction environments.

7 Supplementary Material

The following section provides supplementary details related to the thesis topic. While these insights enrich the context, they are not crucial for understanding the core methodology and contributions.

7.1 Sigmoid Function

The sigmoid function is mathematically defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{7.1}$$

where $z \in (-\infty, +\infty)$ is mapped to an output probability in the range $(0, 1)$.

Applying the sigmoid function to class logits in YOLOv12 [24] offers several distinct advantages for object detection:

- **Independent Binary Classification:** The sigmoid function is different from the softmax function because it allows each class to be evaluated independently, unlike the softmax function, which enforces a mutually exclusive probability distribution (meaning the sum must be equal to 1). In the context of single-class research, it elegantly reduces the classification task to a simple binary question for each proposed bounding box: is there a 'tool' present or not?
- **BCE Loss compatibility:** As the sigmoid function returns an independent probability for each class, it naturally works with the binary cross-entropy (BCE) loss function during training, which is optimized for binary target variables (tool vs background).
- **Probabilistic interpretation:** The model's confidence scores can be treated as true conditional probability of the class given the features, denoted as $P(\text{class}_i \mid \text{features})$, rather than arbitrary scalar values.
- **Stable gradients:** The smooth S-shaped curve of the function helps prevent gradient explosion and vanishing gradient problems during backpropagation.

7.2 Softmax Activation Function

The Softmax function transforms a vector of raw model outputs (logits) into a probability distribution. It ensures that each output value lies between 0 and 1, and that the sum of all the outputs is 1.

For an input vector $\mathbf{z}$ with K elements, the softmax value for each element z_i is defined as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{7.2}$$

where $\sigma(z)_i$ is the predicted probability for class i, and e^{z_i} is the exponential of the input logit, ensuring positive outputs.

Field	**Grinder**	**Tape**
Label	angle grinder	measuring tape
Confidence	high	high
Hazard	yes	no
Hazard Type	sharp object, tripping hazard, electrical cord	–
Severity	high	low
Avoidance	Navigate around the angle grinder with at least 0.5 meters distance to avoid contact.	Proceed with caution, but standard navigation should be sufficient.
Notes	Power tool with cord may complicate safe passage and risk entanglement or tripping.	The object is low-profile and unlikely to obstruct a wheeled robot unless the tape part is extended more.

Table 7.1: Hazard assessment for grinder and measuring tape

Abbreviations

YOLO You Only Look Once

R-ELAN Residual Efficient Layer Aggregation Network

A^2 Area Attention

PSA Partial Spatial Attention

FPN Feature Pyramid Network
[Feature Pyramid Networks]
PAN Path Aggregation Network

PANet Path Aggregation Network

IoU Intersection over Union

CIoU Complete Intersection over Union

NMS Non Maximum Suppression

NN Neural Network
[Neural Networks]
CNN Convolutional Neural Network
[Convolutional Neural Networks]
ML Machine Learning

VLM Vision-Language Model
[Vision-Language Models]
DFL Distribution Focal Loss

BCE Binary Cross-Entropy

CE Cross-Entropy

SGD Stochastic Gradient Descent

TAL Task-Aligned Assignment

LoRA Low-Rank Adaptation

DA2 Depth Anything 2

SAM2 Segment Anything Model 2

List of Figures

List of Tables

Bibliography

[1] L. Xu, Y. Zhang, M. Liu, Y. Li, Y. Li, Y. Yu, Q. Tang, S. Weng, K. Sang, and G. Lin. "Robotics in the Construction Industry: A Bibliometric Review of Recent Trends and Technological Evolution." In: *Applied Sciences* 15.11 (2025). ISSN: 2076-3417. DOI: 10.3390/app15116277.

[2] Y. Ding, M. Zhang, J. Pan, J. Hu, and X. Luo. "Robust object detection in extreme construction conditions." In: *Automation in Construction* 165 (2024), p. 105487. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2024.105487.

[3] M. Nikouei, B. Baroutian, S. Nabavi, F. Taraghi, A. Aghaei, A. Sajedi, and M. E. Moghaddam. "Small object detection: A comprehensive survey on challenges, techniques and real-world applications." In: *Intelligent Systems with Applications* 27 (2025), p. 200561. ISSN: 2667-3053. DOI: https://doi.org/10.1016/j.iswa.2025.200561.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[5] OpenAI. *ChatGPT*. https://chat.openai.com. Large language model used for language editing purposes. 2026.

[6] Grammarly Inc. *Grammarly*. https://www.grammarly.com. AI-powered writing assistant used for grammar and style correction. 2026.

[7] J. Schmidhuber. "Deep learning in neural networks: An overview." In: *Neural Networks* 61 (Jan. 2015), pp. 85–117. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2014.09.003.

[8] Y. LeCun, Y. Bengio, and G. Hinton. "Deep learning." In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539.

[9] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. Available at: http://www.deeplearningbook.org. MIT Press, 2016.

[10] K. O'Shea and R. Nash. *An Introduction to Convolutional Neural Networks*. 2015. arXiv: 1511.08458 [cs.NE].

[11] R. Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[12] S. Song, J. Hong, J. Jeoung, J. Ahn, and T. Hong. "Data-centric enhancement of site-specific automated construction equipment detection in wide-angle site images." In: *Automation in Construction* 179 (2025), p. 106483. DOI: 10.1016/j.autcon.2025.106483.

[13] J. S. Ryu, H. Kang, Y. Chu, and S. Yang. "Vision-language foundation models for medical imaging: a review of current practices and innovations." In: *Biomedical Engineering Letters* 15.5 (2025), pp. 809–830. DOI: 10.1007/s13534-025-00484-6.

[14] L. T. Ramos and A. D. Sappa. "A Decade of You Only Look Once (YOLO) for Object Detection: A Review." In: *IEEE Access* 13 (2025), pp. 192747–192794. DOI: 10.1109/ACCESS.2025.3630988.

[15] J. Redmon and A. Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV].

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[17] J. Redmon and A. Farhadi. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[18] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. *Path Aggregation Network for Instance Segmentation*. 2018. arXiv: 1803.01534 [cs.CV].

[19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020. arXiv: 2004.10934 [cs.CV].

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[21] R. Khanam and M. Hussain. *What is YOLOv5: A deep look into the internal features of the popular object detector*. 2024. arXiv: 2407.20892 [cs.CV].

[22] A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. *YOLOv10: Real-Time End-to-End Object Detection*. 2024. arXiv: 2405.14458 [cs.CV].

[23] R. Khanam and M. Hussain. *YOLOv11: An Overview of the Key Architectural Enhancements*. 2024. arXiv: 2410.17725 [cs.CV].

[24] Y. Tian, Q. Ye, and D. Doermann. *YOLOv12: Attention-Centric Real-Time Object Detectors*. 2025. arXiv: 2502.12524 [cs.CV].

[25] Y. Li, L. Kaiser, S. Bengio, and S. Si. *Area Attention*. 2020. arXiv: 1810.10126 [cs.LG].

[26] Y. Tian, Q. Ye, and D. Doermann. *YOLOv12: Attention-Centric Real-Time Object Detectors*. Version 1.0.0. GitHub repository. 2025.

[27] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[28] N. Deluxni and P. Sudhakaran. "Underwater debris detection using YOLOv12 with enhanced feature extraction using R-ELAN and FlashAttention network." In: *Results in Engineering* 28 (2025), p. 107282. ISSN: 2590-1230. DOI: https://doi.org/10.1016/j.rineng.2025.107282.

[29] C. Feng, Y. Zhong, Y. Gao, M. R. Scott, and W. Huang. *TOOD: Task-aligned One-stage Object Detection*. 2021. arXiv: 2108.07755 [cs.CV].

[30] J. Hosang, R. Benenson, and B. Schiele. *Learning non-maximum suppression*. 2017. arXiv: 1705.02950 [cs.CV].

[31] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer. *SAM 2: Segment Anything in Images and Videos*. 2024. arXiv: 2408.00714 [cs.CV].

[32] H. Li, W. Zheng, J. He, Y. Liu, X. Lin, X. Yang, Y.-C. Chen, and C. Guo. *DA^2: Depth Anything in Any Direction*. 2025. arXiv: 2509.26618 [cs.CV].

[33] Q. Chen and X. Yin. "Tailored vision-language framework for automated hazard identification and report generation in construction sites." In: *Advanced Engineering Informatics* 66 (2025), p. 103478. ISSN: 1474-0346. DOI: https://doi.org/10.1016/j.aei.2025.103478.

[34] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].

[35] OortDataHub. *Diverse Tools Image Dataset for Machine Learning*. https://www.kaggle.com/datasets/oortdatahub/diverse-tools-image-dataset-for-machine-learning. Accessed: 2026-03-03.

[36] Kaleido AI. *Remove.bg*. https://www.remove.bg. AI-powered background removal tool.

[37] Canva Pty Ltd. *Canva*. https://www.canva.com. Graphic design platform.

[38] Roboflow. *Roboflow (Version 1.0)*. https://roboflow.com. Computer vision management platform.

[39] OpenAI. *OpenAI API Reference: Responses Endpoint*. https://platform.openai.com/docs/api-reference/responses. Accessed: 2026-03-09.

[40] aswin00000. *ConstructionSiteCleanedDataSet*. `https://huggingface.co/datasets/aswin00000/ConstructionSiteCleanedDataSet`.

[41] Kaggle Inc. *Kaggle: A Data Science and Machine Learning Platform*. `https://www.kaggle.com/`. Cloud computing services used for model training.