UNIVERSITY OF SCIENCE AND TECHNOLOGY – DANANG UNIVERSITY

FACULTY OF PROJECT MANAGEMENT

GRADUATION PROJECT

MAJOR: INDUSTRIAL MANAGEMENT

TITLE OF THESIS:

AUTOMATED DEFECT DETECTION FOR METAL SURFACES

Supervior: Dr. Nguyễn Thị Phương Quyên

Student: Huỳnh Thị Thúy

Student ID: 118210204

Class: 21QLCN2

Danang City, June 2025

UNIVERSITY OF SCIENCE AND TECHNOLOGY **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

FACULTY OF PROJECT MANAGEMENT Độc lập – Tự do – Hạnh phúc

GRADUATION PROJECT ASSIGNMENT

Student Name: Huỳnh Thị Thúy **Student ID:** 118210204 **Class:** 21QLCN2

Faculty: Project Management **Major:** Industrial Management

1. **Project Title:** *Automated Defect Detection for Metal Surfaces*
2. **Field of Study:** ☐ There is an intellectual property agreement signed to carry out the results
3. **Initial Data and Figures:**

- Dataset: 871 real-world defect images from Quoc Quang Electro-Acoustic Co., Ltd.
- Training set: 763 images
- Validation set: 108 images
- Defect classes: crack, pitted, patches
- Tools: YOLOv11, Gradio, Python, Google Colab

4. **Content Outline and Calculations:**

- Chapter 1 – Introduction
- Chapter 2 – Theoretical Basis
- Chapter 3 – Current Situation Analysis
- Chapter 4 – Methodology
- Chapter 5 – Experimental Results and Analysis
- Chapter 6 – Conclusion

5. **Supervisor's Name:** *Dr. Nguyễn Thị Phương Quyên*
6. **Assignment Date:** *March 2025*
7. **Completion Date:** *June 2025*

Danang City, June 2025

Head of department **Supervior**

ACKNOWLEDGEMENT

Upon completing this report, the author would like to express her sincere gratitude to Dr. Nguyễn Thị Phương Quyên, the Supervisor, for her dedicated guidance, valuable feedback, and enthusiastic support throughout the process of completing this graduation thesis.

The author also wishes to extend her appreciation to the members of the Evaluation Committee for their time, insightful comments, and constructive feedback, which are vital for the improvement of this work.

Due to the limited time and the author's insufficient experience, this report may still contain shortcomings and errors. Therefore, the author respectfully looks forward to receiving further suggestions and evaluations from both the Supervisor and the Committee to enhance the quality of future research and academic development.

Danang City, June 2025

Student

Huỳnh Thị Thúy

DECLARATION

I hereby declare that this graduation thesis is my own original work, carried out under the supervision of Dr. Nguyễn Thị Phương Quyên. All the data, results, figures, and references presented in this report are truthful, properly cited, and have not been previously submitted for any other academic purpose.

I take full responsibility for the honesty and integrity of the content presented herein.

Danang, June 2025
Author

Huỳnh Thị Thúy

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background and Motivation

In today's audio electronics manufacturing industry, maintaining high quality standards is essential for meeting international requirements and customer expectations. This is particularly important for companies like Quoc Quang Electro-Acoustic Co., Ltd. (GGEC Vietnam), which serves as a key manufacturing partner for globally recognized brands including JBL, Harman Kardon, and Sony. The growing demand for high-quality audio devices requires greater precision in component manufacturing, where even small surface defects in metal parts such as crack, pitted and patches can significantly affect both appearance and performance.

Quality control in modern manufacturing has moved from simple inspection methods to advanced defect prevention strategies. However, despite having modern manufacturing equipment, GGEC Vietnam's current Incoming Quality Control (IQC) process still relies heavily on traditional manual visual inspection. While this approach works reasonably well in controlled conditions, it shows significant limitations when dealing with high-volume production typical of modern manufacturing.

The combination of artificial intelligence, computer vision technology, and Industry 4.0 principles offers a great opportunity to improve traditional quality control methods. This research studies the development of an advanced AI-based visual inspection system, using state-of-the-art deep learning models—specifically the YOLOv11 object detection system—to create a reliable, scalable, and intelligent defect detection system for metal surface problems.

1.2 Problem Statement and Research Gap

1.2.1 Problem Statement

Current manufacturing quality control processes in the audio electronics industry face several challenges that significantly impact operational efficiency and product quality. The existing manual inspection system at GGEC Vietnam shows these common limitations:

Operational problems: current inspection methods require about 2-3 minutes per component, resulting in over 100 labor-hours weekly dedicated only to quality control activities. This resource use represents a major operational bottleneck that increases directly with production volume.

Inconsistent detection results: human-based inspection processes naturally vary between workers, with disagreement between inspectors happening in about 12% of borderline cases. Additionally, analysis shows that 7-10% of small defects—including micro-crack, early pitting, and surface patches—are regularly missed during routine inspections.

Late defect detection: the current inspection method often fails to identify defects until later production stages, causing significant costs including material waste, rework operations, and potential warranty issues.

Limited scalability: as production volumes increase to meet global demand, the manual inspection approach becomes increasingly difficult to maintain, creating potential quality bottlenecks that could affect delivery schedules and customer satisfaction.

1.2.2 Research Gap Identification

Despite significant advances in computer vision and deep learning technologies, several research gaps exist in the application of these technologies to industrial quality control:

Limited availability of large-scale real-world industrial datasets: While this project utilizes real defect images collected from an actual audio electronics manufacturing company, the overall availability of large, diverse, and annotated datasets from real industrial environments remains limited. Most previous research still relies on synthetic or controlled lab data, which may not fully reflect real-world complexity.

Insufficient performance studies of latest models: While YOLO models have been widely used for general object detection, limited studies have evaluated the performance of the newest YOLOv11 architecture specifically for industrial surface defect detection in real manufacturing conditions.

Gap in practical implementation: Most academic research focuses on algorithm development but lacks practical implementation frameworks that can be directly deployed in industrial settings with user-friendly interfaces.

Industry-specific optimization: Current research often provides general solutions but lacks optimization for specific industries like audio electronics manufacturing, where defect types and requirements are unique.

These challenges and research gaps together represent an important opportunity for applying advanced computer vision technologies to real-world industrial quality control situations, particularly in the specialized area of audio electronics manufacturing.

1.3 Research Objectives

This study aims to develop and test an intelligent defect detection system capable of automatically identifying and classifying surface problems in metal components. The research objectives are organized as follows:

Main objective: To design, build, and evaluate a comprehensive AI-powered visual inspection system for automated detection of surface defects in metal components used in audio electronics manufacturing.

Specific Objectives:

- Defect classification and detection: to develop a strong classification system capable of identifying and categorizing three critical defect types—crack, pitted surfaces, patches—using real data from actual production environments.
- Deep learning model development: to implement and optimize a YOLOv11-based deep learning model specifically designed for industrial surface inspection applications, using advanced transfer learning techniques and industry-specific improvements.
- Dataset creation and model training: to collect, annotate, and use a comprehensive dataset of 871 high-resolution images of defective components, ensuring statistical significance and model reliability through rigorous training and validation processes.
- Application development and user interface: to develop an easy-to-use, operator-friendly application interface enabling real-time image upload, processing, and defect visualization with comprehensive reporting capabilities.
- Performance analysis and improvement: to conduct thorough performance evaluation using industry-standard metrics including precision, recall, mean average precision (map@0.5 and map@0.5:0.95), and to identify improvement opportunities for better accuracy and computational efficiency.

1.4 Scope and Limitations

1.4.1 Research Scope

Application area: this research specifically focuses on incoming raw material inspection processes within speaker manufacturing operations, concentrating only on external surface defect identification and classification.

Dataset information: the research foundation consists of 871 high-resolution digital images of defective metal components, systematically collected from actual production environments at GGEC Vietnam. Each image includes precise bounding-box annotations for three distinct defect types.

Technology framework: the study uses YOLOv11 architecture, chosen for its proven excellent performance in real-time object detection, computational efficiency suitable for industrial deployment, and demonstrated effectiveness in small-object detection scenarios.

Prototype development: the research includes a functional prototype application using the gradio framework, providing end-users with streamlined defect analysis capabilities through simple image upload and automated annotation features.

1.4.2 Research Limitations

Excluded elements: this study does not include internal material defect analysis, three-dimensional geometric evaluation, or full real-time video stream processing integration. These elements are identified as potential areas for future research development.

Time constraints: the study focuses on static image analysis rather than dynamic inspection scenarios, though the basic framework supports future extension to continuous monitoring applications.

Dataset limitations: although the dataset is based on real-world images collected from industrial processes, the number of available images is limited. This may impact the model's generalization to less common defect types.

Scope of defect types: this study focuses on three main surface defect classes such as crack, pitted, patches based on their frequency and visual clarity. Other defect types such as deformation, corrosion, or discoloration are not addressed and remain open for future exploration

1.5 Research Contributions and Significance

This study delivers several important contributions to both academic knowledge and industrial practice:

1.5.1 Academic Contributions

New dataset development: Creation of a comprehensive, industry-validated dataset containing real-world defect samples from operational manufacturing environments. This

dataset addresses an important gap in publicly available, annotated industrial defect datasets and provides a valuable resource for future research projects.

Advanced AI model application: First application of YOLOv11 architecture within the specialized area of metal surface defect detection for audio electronics manufacturing, contributing to understanding of state-of-the-art object detection models in precision manufacturing contexts.

Comprehensive performance testing: Establishment of rigorous performance evaluation methods and baseline metrics for industrial defect detection applications, providing empirical foundations for future comparative studies.

1.5.2 Industrial Contributions

Practical automation solution: development of a deployable, production-ready inspection system demonstrating clear pathways for moving from manual to automated quality control processes in industrial environments.

Industry 4.0 integration: alignment with modern smart manufacturing principles, offering scalable, traceable, and data-driven solutions for improved productivity and operational decision-making capabilities.

Economic impact potential: demonstration of significant cost reduction opportunities through automation, with implications for improved resource allocation and competitive advantage in global markets.

1.6 Thesis Organization and Structure

This thesis is systematically organized into six comprehensive chapters, each contributing essential elements to the overall research story:

Chapter 1 – Introduction: Establishes the research foundation through comprehensive background analysis, problem identification, objective development, scope definition, methodology framework, and contribution description.

Chapter 2 – Theoretical Basis: Provides thorough examination of relevant academic literature, technology developments, and theoretical foundations in computer vision, defect detection methods, and YOLO-based architectures for industrial quality control applications.

Chapter 3 – Current Situation Analysis: Conducts detailed analysis of existing manual inspection processes at GGEC Vietnam, systematically identifying operational limitations,

quantifying consequences, and establishing technical requirements for automation implementation.

Chapter 4 – Methodology: Presents comprehensive description of the end-to-end research approach, including dataset preparation methods, model architecture design, training strategies, evaluation procedures, and application development frameworks.

Chapter 5 – Experimental Results and Analysis: Details empirical results of model training and validation processes, presents comprehensive performance metrics analysis, discusses practical implications.

Chapter 6 – Conclusion, and provides evidence-based recommendations for industrial implementation and future research directions.

CHAPTER 2: THEORETICAL BASIS

To provide a high-level overview of the system's theoretical foundation, the figure below summarizes the key components discussed throughout this chapter. It includes deep learning classification methods, object detection concepts, YOLOv11 architecture, performance metrics, and deployment via a web-based interface. This visual map supports readers in understanding how all elements are integrated into a cohesive defect detection pipeline.

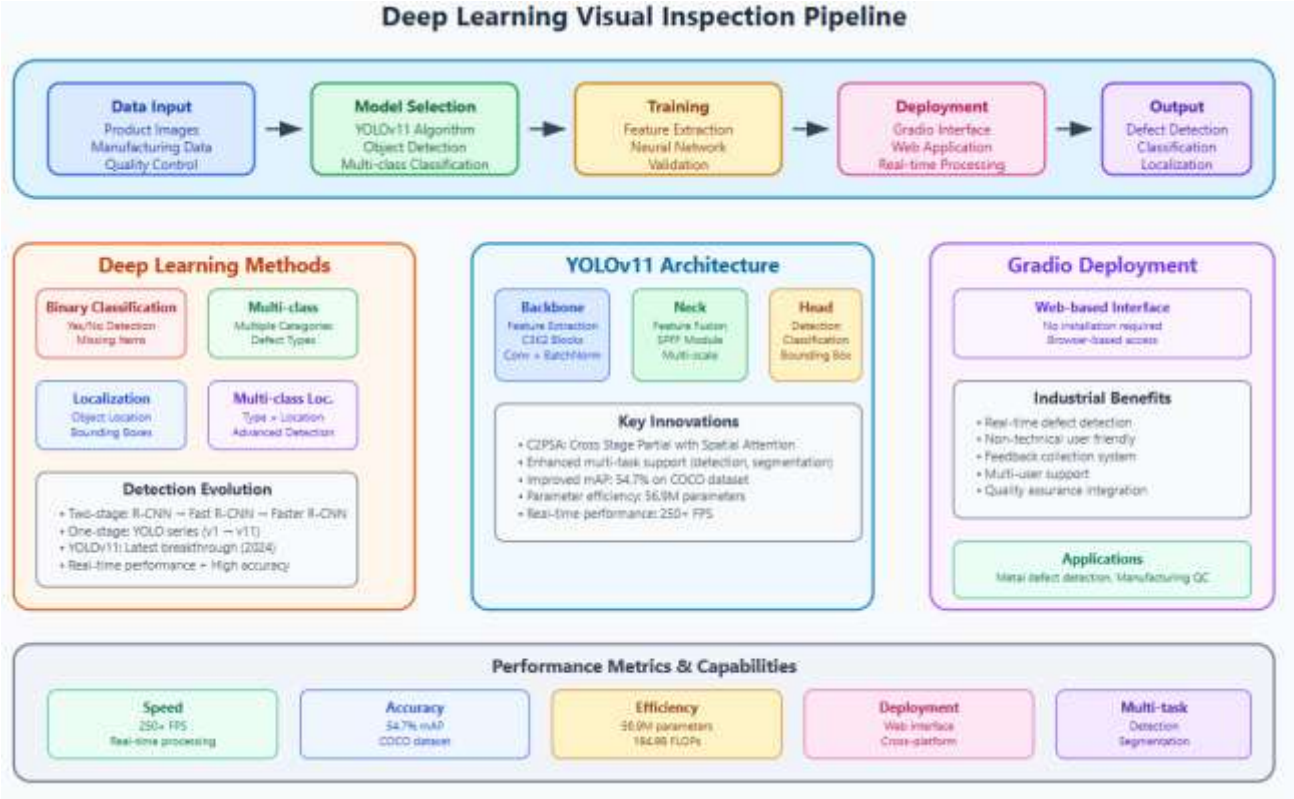

Figure 2.1 Overview of Deep Learning Methods, YOLOv11 Architecture, and Deployment Pipeline

2.1 Deep learning for visual inspection

Deep learning approaches use artificial neural networks that work on the principles of a human brain. The networks are interconnected layers of neurons. Each neuron performs a mathematical calculation to analyze data, identify patterns, and generate a prediction. [1]

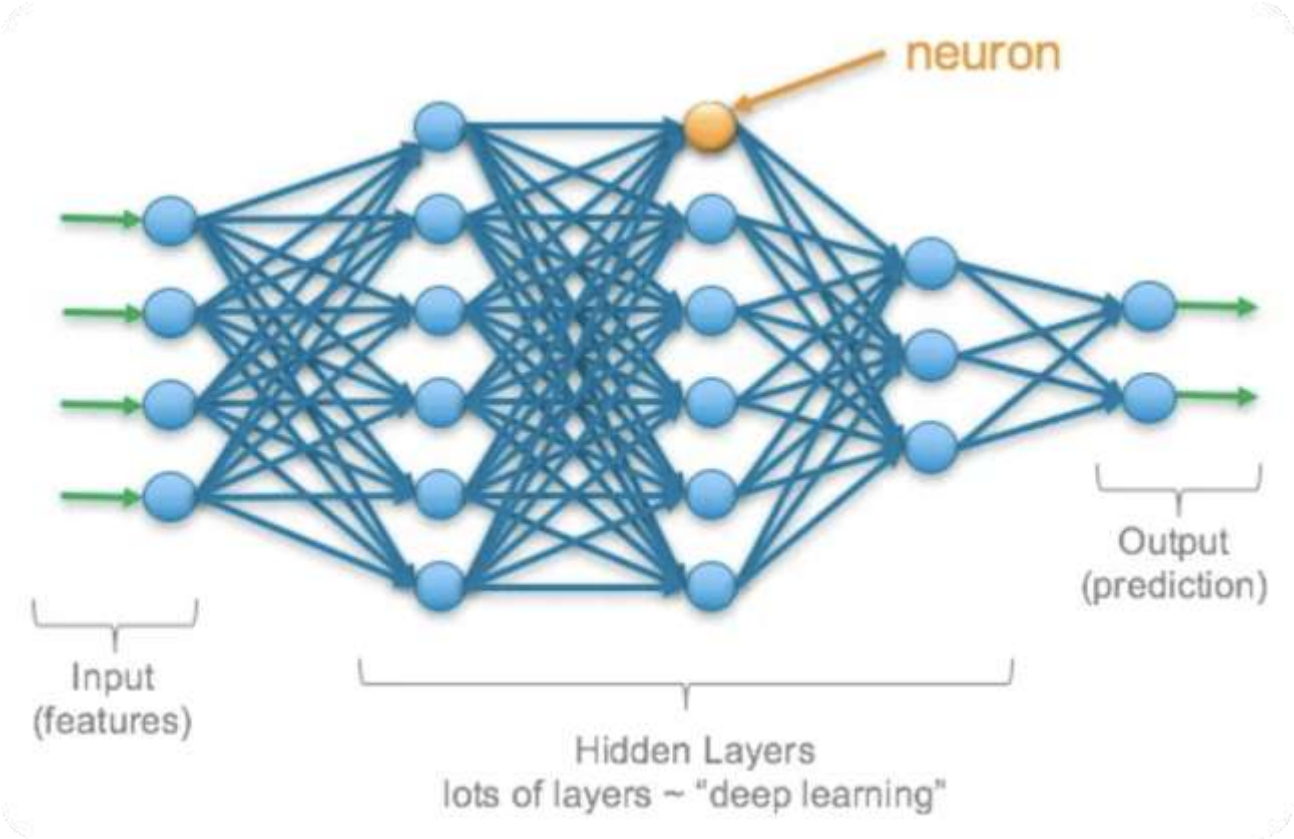

Figure 1.2 Artificial neural network. [2]

In quality inspection, deep learning models include computer vision frameworks that automatically learn and extract features from product images.

Developing computer vision models requires experts to train a neural network on relevant datasets and run validations on a new dataset to check performance.

Once validated, experts can deploy these models on cameras and sensors using various deployment tools such as PyTorch, ONNX, and OpenVINO.

Vision-based quality inspection uses multiple methods to detect and localize damages, cracks, and missing items. The list below mentions four modern deep learning approaches.

2.1.1 Binary classification for visual inspection

Binary classification refers to the task of categorizing images into one of two classes, such as determining whether or not a defect is present in an object.

Based on visual data, a classification model outputs a binary yes/no decision. They help detect missing items. For example, a classification model can detect whether an item is missing or not in a product.

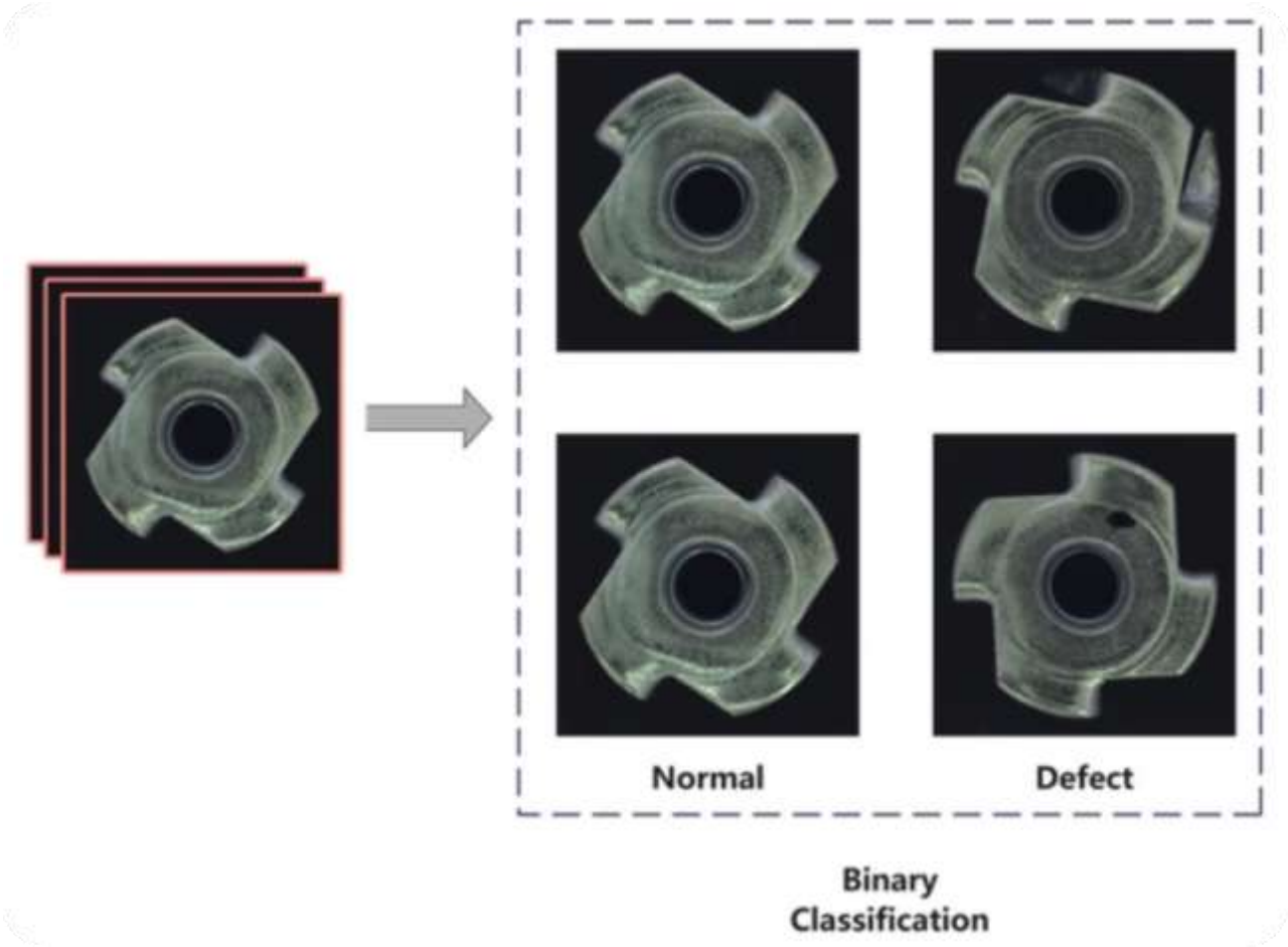

Figure 2.3 Binary classification of a car part. [2]

2.1.2. Multi-class classification

Multi-class classification is the task of categorizing images into more than two classes. It assigns each image to one of several predefined categories.

For example, a multi-class classification model may analyze a product's image and return probabilities for multiple damage or crack types, indicating which one is most likely present.

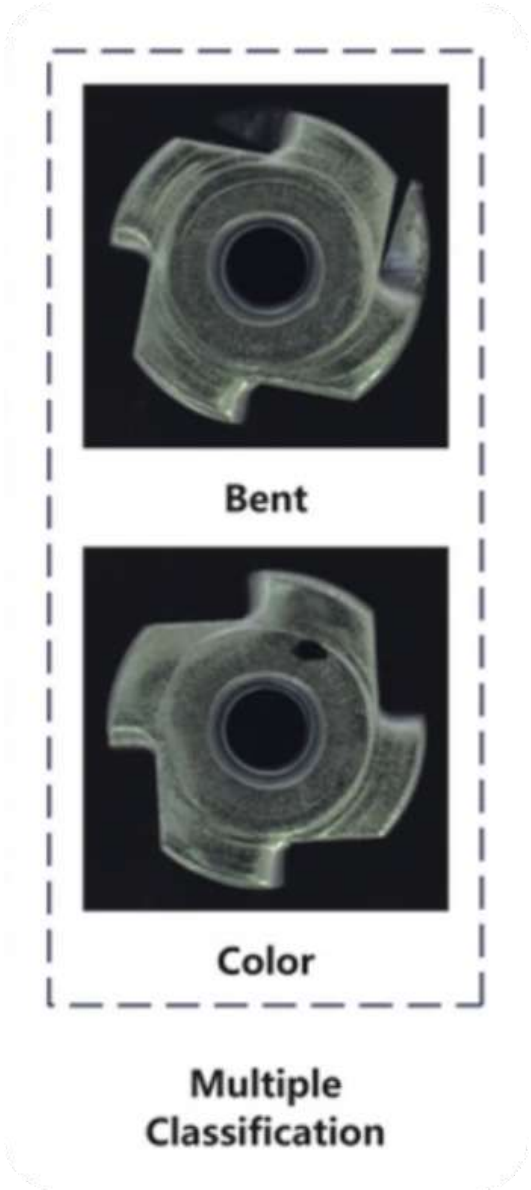

Figure 2.4 Multi-class classification detecting bent and color. [2]

This is useful in manufacturing where various defects, such as crack or patches, might require different handling procedures.

2.1.3. Object Localization and Multi-class Detection

Object localization refers to identifying the specific position of one or more objects or features within an image. In defect detection, this involves using object detection models to predict bounding boxes or coordinates that highlight the region of damage. This is particularly

useful in industrial applications like crack detection on metal parts, where knowing the exact location enables targeted repairs.

When multiple types of defects need to be identified and located simultaneously, the task becomes **multi-class localization**. This advanced form of object detection not only determines the position of each defect but also classifies it into predefined categories—such as *pitted*, *patches*, or *cracks*. Multi-class localization provides detailed insights into both the type and location of each defect, supporting more effective and automated quality control.

Figure 2.5 Model localizing a pitted

Figure 2.6 Model locating multiple defect types.

2.2. Overview of Object Detection Models

2.2.1. Introduction

Object detection is a fundamental computer vision task that combines object localization and classification. Unlike image classification which only identifies what objects are present in an image, object detection determines both what objects exist and where they are located by predicting bounding boxes and class labels simultaneously.

The evolution of object detection has been marked by significant architectural innovations, progressing from traditional computer vision approaches to modern deep learning-based methods. Contemporary object detection models can be broadly categorized into two main paradigms: two-stage and one-stage detectors.

2.2.2 Two-Stage Detection Models

Two-stage detectors follow a sequential approach where region proposal generation precedes classification and localization refinement. This methodology prioritizes accuracy through careful candidate selection and processing.

R-CNN Family Evolution

R-CNN (2014) introduced the concept of applying CNNs to region proposals generated by selective search. Despite achieving significant accuracy improvements, the model suffered from computational inefficiency, requiring approximately 47 seconds per image due to individual CNN forward passes for each proposal. [2]

Fast R-CNN (2015) addressed the computational bottleneck by introducing feature sharing through RoI pooling. By extracting features once for the entire image and then pooling region-specific features, inference time was reduced to 0.5 seconds per image while maintaining detection accuracy.

Faster R-CNN (2015) eliminated the selective search dependency by introducing the Region Proposal Network (RPN), creating the first fully neural end-to-end object detection system. The RPN generates object proposals using anchor boxes at multiple scales and aspect ratios, achieving approximately 5 FPS while improving detection performance. [3]

Characteristics of Two-Stage Models

Two-stage detectors excel in scenarios requiring high precision, particularly for small object detection and complex scenes. However, their multi-step processing pipeline inherently limits inference speed, making them less suitable for real-time applications.

2.2.3. One-Stage Detection Models

One-stage detectors revolutionized object detection by formulating it as a single regression problem, directly predicting bounding boxes and class probabilities from input images without explicit region proposal generation.

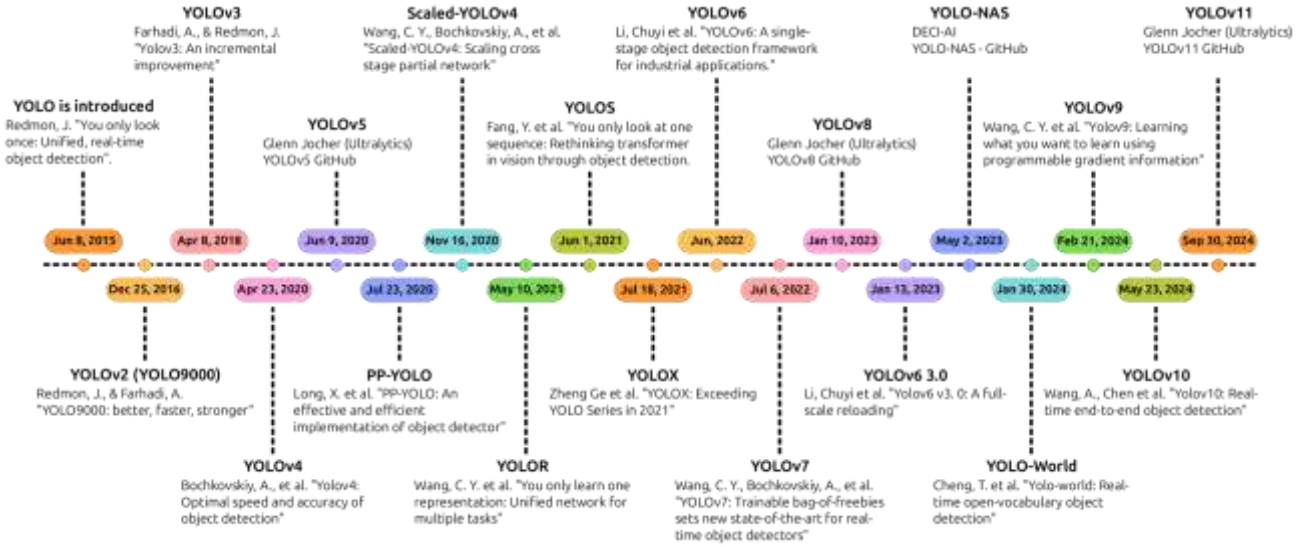

Figure 2.7 Milestones in YOLO Evolution (V1 to V11) [4]

YOLO Architecture Evolution

YOLOv1 (2015) pioneered the one-stage approach by dividing input images into a 7×7 grid, with each cell responsible for predicting bounding boxes and class probabilities for objects whose centers fall within the cell. This approach achieved 45 FPS but struggled with small object detection.

YOLOv2/YOLO9000 (2016) introduced anchor boxes and batch normalization, improving both accuracy and training stability. YOLO9000 demonstrated the model's scalability by detecting over 9,000 object classes through joint training on detection and classification datasets.

YOLOv3 (2018) incorporated multi-scale feature pyramids with predictions at three different scales (13×13, 26×26, 52×52), significantly improving small object detection while maintaining real-time performance.

YOLOv4 (2020) achieved state-of-the-art performance by integrating numerous training and architectural improvements, including CSPDarknet53 backbone, SPP and PAN neck architectures, and advanced data augmentation techniques like Mosaic augmentation.

YOLOv5-YOLOv8 (2020-2023) focused on deployment optimization and architectural refinements, with YOLOv8 adopting an anchor-free design and supporting multiple tasks including detection, segmentation, and pose estimation.

YOLOv11 (2024) represents the latest evolution with enhanced efficiency for edge deployment and industrial applications, achieving over 250 FPS on modern hardware while maintaining competitive accuracy.

Other Notable One-Stage Models

SSD (Single Shot MultiBox Detector) employs multi-scale feature maps from different layers of the backbone network, enabling detection of objects at various scales within a single forward pass.

RetinaNet addressed the class imbalance problem in one-stage detectors through the introduction of Focal Loss, which down-weights easy examples and focuses training on hard examples.

2.3 YOLOv11 Algorithm and Application

2.3.1 Introduction and Key Innovations

Released by the Ultralytics team on 30 September 2024, YOLOv11 represents the latest breakthrough in real-time object detection technology. This version supports various tasks, such as detecting objects, segmenting images, classifying images, estimating poses, and detecting objects with specific orientations. The algorithm improves how features are identified by refining its core components and increasing processing speed through better design and training methods. Notably, YOLOv11 achieves a higher mean average precision (mAP) on the COCO dataset while utilizing fewer computational resources than its predecessors. These updates not only advance object detection but also make it easier for developers to use, offering a strong foundation for future projects and applications. YOLOv11 (2024): The latest iteration, YOLOv11, was designed for state-of-the art performance across tasks such as object detection, instance segmentation, and pose estimation. Its main innovations include the following:

(1) Multi-task processing: Supports diverse tasks, including instance segmentation and pose estimation, making it adaptable to complex scenarios.

(2) Architecture optimization: Incorporates self-attention mechanisms and transformer-based designs, improving precision and efficiency, particularly in complex and multi-object scenarios.

(3) Robustness in challenging conditions: Demonstrated exceptional performance in detecting small objects and handling occlusions, particularly in low-contrast and densely populated environments.

(4) Enhanced generalization: Leverag ing novel data augmentation and transfer learning techniques, YOLOv11 exhibits improved performance across datasets and domains such as remote sensing and medical imaging. [4]

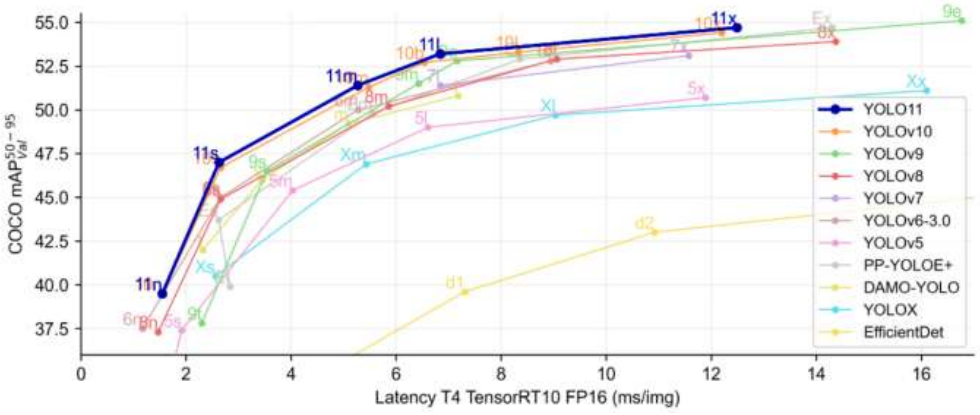

Figure 2.8 Comparison of accuracy and inference delay performance of YOLO series models on the COCO dataset [6]

YOLOv11 introduces an ultra-lightweight design, achieving significantly improved speed and efficiency compared to its predecessors [6]. Figure 8 illustrates the comparative performance of YOLOv11 against earlier YOLO versions, highlighting differences in accuracy and speed. The X-axis depicts image processing latency (in milliseconds) on a T4 TensorRT10 FP16 system, while the Y-axis represents mean average precision (mAP) on the COCO dataset, serving as a measure of detection accuracy. Models closer to the top-left corner (low latency and high accuracy) are better suited for real-time applications. YOLOv11 achieves an excellent balance between accuracy and speed, with model vari ants ranging from nano to xlarge, making it highly competitive for practical use. Table 1 compares the top-performing "x" versions of different YOLO models in terms of input resolution (640 pixels), mAP50-95 (COCO dataset), parameter count, and computational cost (FLOPs). Key highlights include the following:

(1) Accuracy: YOLOv11x achieves an mAP of 54.7%, outperforming YOLOv10-x (+0.3%), YOLOv8x (+0.8%), and YOLOv7-x (+1.6%). While YOLOv9e achieves slightly higher accuracy (55.6%), it requires significantly more parameters and FLOPs, making YOLOv11x more efficient.

(2) Parameter efficiency: YOLOv11x uses 56.9M parameters, representing a 41.5% reduction compared to YOLOv11xu (97.2 M) and a 16.6% reduction compared to YOLOv8x (68.2 M). Although YOLOv11x has slightly more parameters than YOLOv10-x (29.5 M), the precision gain (+0.3%) justi f ies the trade-off.

(3) Computational cost: YOLOv11x requires 194.9 B FLOPs, reducing computational demands by 20.9% compared to YOLOv11xu (246.4 B) and 24.4% compared to YOLOv8x (257.8 B). While YOLOv11x incurs slightly higher computational costs than YOLOv10-x (+2.4 B FLOPs), it offers a +0.3% precision gain, highlighting its efficiency. In summary, YOLOv11x combines higher accuracy with reduced resource consumption, offering superior efficiency compared to earlier YOLO models. Even when compared to YOLOv9e (which achieves slightly higher accuracy), YOLOv11x significantly reduces parameter count (−65%) and FLOPs, achieving a better balance between performance and efficiency. Although YOLOv7-x has lower computational demands, its lower accuracy (53.1% mAP) limits its practicality compared to YOLOv11x.

Table 2.1 Performance comparison of the YOLO series

Model	Size (Pixels)	mAP^{val}_{50-95}	Params	FLOPs
YOLOv5xu	640	53.2	97.2 M	246.4 B
YOLOv7-x	640	53.1	71.3 M	189.9 G
YOLOv8x	640	53.9	68.2 M	257.8 B
YOLOv9e	640	55.6	58.1 M	192.5 B
YOLOv10-x	640	54.4	29.5 M	160.4G
YOLO11x	640	54.7	56.9 M	194.9B

2.3.2 Comparison of YOLOv11 with Other Algorithms

Recent advancements in deep learning have significantly enhanced the development of object detection and image segmentation algorithms [6]. Each of these algorithms—such as YOLOv11, EfficientDet, Mask R-CNN, U-Net, RT-DETR, and ViT-CoMer—offers unique strengths and limitations, making them suitable for different tasks and application domains.

Table 2 provides a comparative overview of these state-of-the-art models, outlining their typical use cases, core advantages, and known limitations. This comparative analysis serves as a valuable reference for researchers and practitioners, offering practical insights into the effectiveness and applicability of various models in diverse real-world scenarios.

Table 1.2 Comparison table of YOLOv11 and other algorithms. [7]

Algorithm	**Advantages**	**Limitations**
YOLOv11	High precision and speed; Multi-task support; Efficient architecture.	Increased complexity due to multi-task capabilities.
EfficientDet	Lightweight and efficient; Adaptable to resource-constrained devices;	Requires fine-tuned hyperparameters; Limited support for tasks beyond detection;
MaskR-CNN	High accuracy insegmentation; Flexible and extensible;	Slower inference speed; Complex architecture;
U-Net	Simple and specialized for segmentation; Excels in medical imaging;	Not suitable for detection tasks; Limited generalization to complex scenarios;
RT-DETR	Real-time detection; No post processing required;	Limited real-world applications due to being a new approach;
ViT-CoMer	Combines convolution and multi-scale features; Strong performance in dense prediction tasks;	High computational demand and hardware requirements;

YOLOv11 emerges as a highly advanced algorithm for object detection and segmentation, offering a well-balanced combination of precision, real-time processing, and multi-task capabilities. Its ability to operate efficiently in complex environments makes it suitable for a wide range of tasks, including object detection, instance segmentation, and pose estimation.

In the context of construction site recognition, YOLOv11 proves especially effective due to its high accuracy and ability to process visual data in real time. It can accurately detect and segment multiple targets—such as bulldozers, cranes, and workers—even under dynamic and cluttered conditions, which enhances its adaptability and practical utility.

Moreover, the model's optimized architecture and computational efficiency allow it to be deployed on resource-constrained devices, making it well-suited for real-world engineering

and industrial applications. For these reasons, this study adopts YOLOv11 as the core algorithm to explore its effectiveness in intelligent construction site monitoring, with the goal of supporting safer site management, enabling automated surveillance, and laying a theoretical foundation for future research and innovation in related fields.

2.3.3 Network Structure

As illustrated in Figure 9, YOLOv11 retains the fundamental architectural framework of the YOLO series, consisting of three primary components: the **Backbone**, **Neck**, and **Head**. The **Backbone** serves as the core feature extractor, responsible for capturing essential patterns, textures, and spatial structures from the input image. This stage lays the groundwork for identifying object-relevant regions. Next, the **Neck** operates as a multi-scale feature fusion module. It aggregates feature maps from different stages of the network, enabling the model to handle objects of varying sizes more effectively and improving contextual awareness. Finally, the **Head** is responsible for generating the final detection outputs, which include object class probabilities and bounding box coordinates. This component translates learned features into actionable predictions. Together, these modules form an efficient and streamlined architecture that enables YOLOv11 to achieve both high detection accuracy and real-time performance. By optimizing the processes of feature extraction, fusion, and output generation, YOLOv11 proves well-suited for complex object detection tasks in dynamic environments.

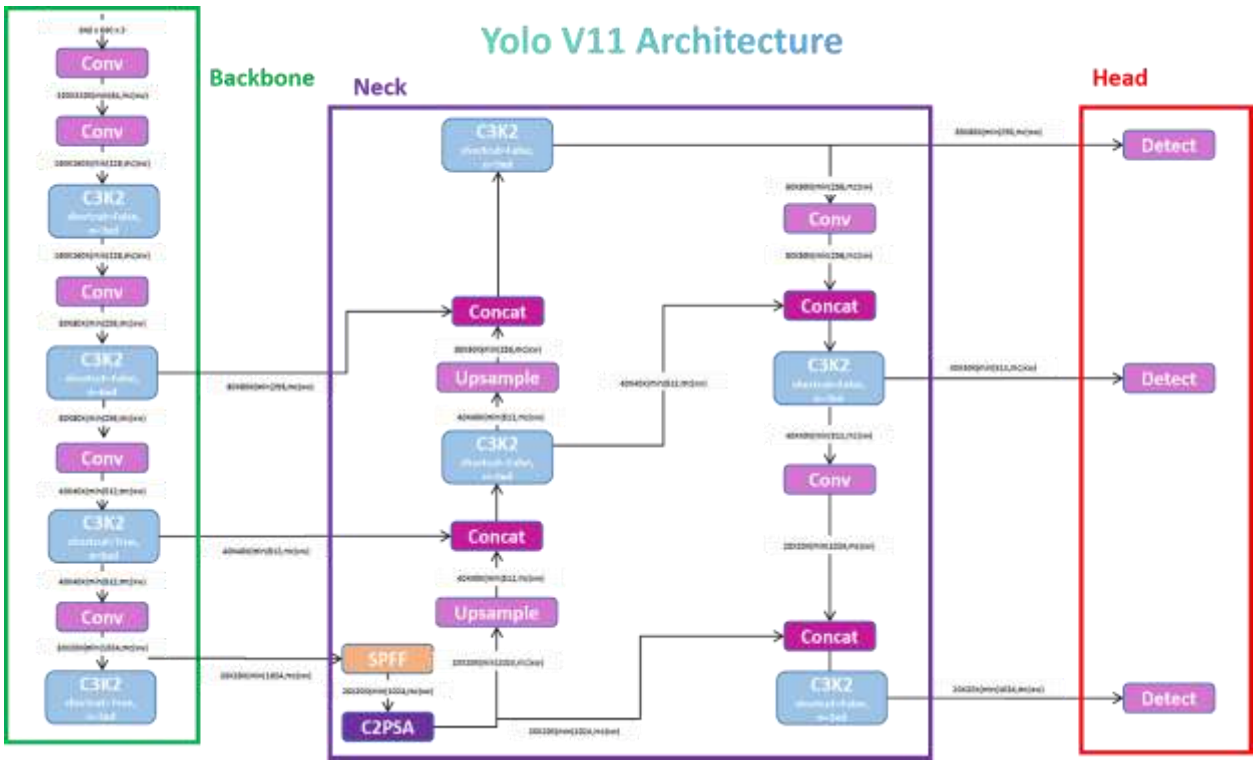

Figure 2.9 YOLOv11 Model Architecture [9]

The main architectural innovations in YOLOv11 revolve around the **C3K2 block**, the **SPFF module**, and the **C2PSA block**, all of which enhance its ability to process spatial information while maintaining high-speed inference.

2.3.3.1 Backbone

a. Convolutional Block

This block is named as Conv Block which process the given c,h,w passing through a 2D convolutional layer following with a 2D Batch Normalization layer at last with a SiLU Activation Function.

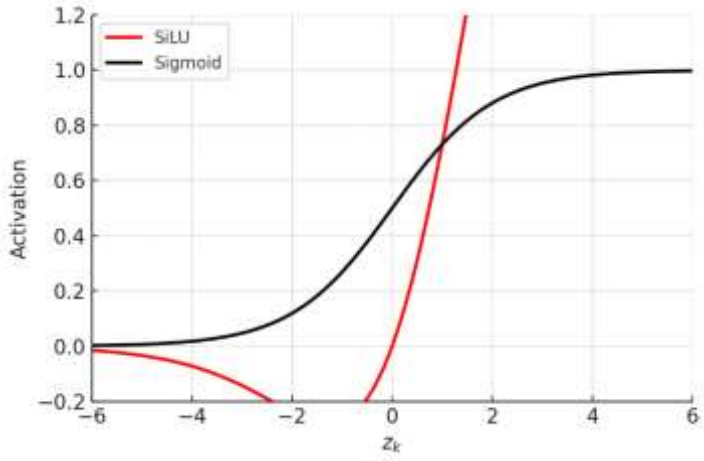

Figure 2. SiLU vs Sigmoid Activation Function [9]

b. Bottle Neck

This is a sequence of convolutional block with a shortcut parameter, this would decide if you want to get the residual part or not. It is similar to the ResNet Block, if shortcut is set to False then no residual would be considered.

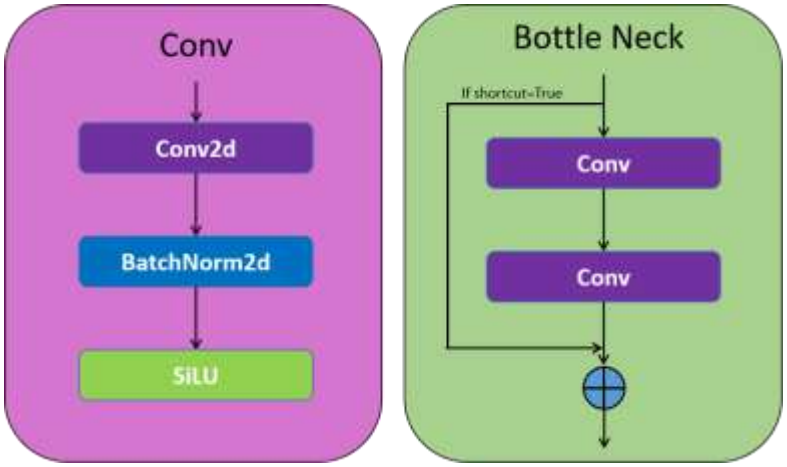

Figure 2.11 Convolutional Block and Bottle Neck Layer [9]

c. C2F

The C2F block (Cross Stage Partial Focus, CSP-Focus), is derived from CSP network, specifically focusing on efficiency and feature map preservation. This block contains a Conv Block then splitting the output into two halves (where the channels gets divided), and they are processed through a series of 'n' Bottle Neck layers and lastly concatinates every layer output following with a final Conv block. This helps to enhance feature map connections without redundant information.

d. C3K2

YOLOv11 uses C3K2 blocks to handle feature extraction at different stages of the backbone. The smaller 3x3 kernels allow for more efficient computation while retaining the model's ability to capture essential features in the image. At the heart of YOLOv11's backbone is the C3K2 block, which is an evolution of the CSP (Cross Stage Partial) bottleneck introduced in earlier versions. The C3K2 block optimizes the flow of information through the network by splitting the feature map and applying a series of smaller kernel convolutions (3x3), which are faster and computationally cheaper than larger kernel convolutions. By processing smaller, separate feature maps and merging them after several convolutions, the C3K2 block improves feature representation with fewer parameters compared to YOLOv8's C2f blocks.

The C3K block contains a similar structure to C2F block but no splitting will be done here, the input is passed through a Conv block following with a series of 'n' Bottle Neck layers with concatinations and ends with final Conv Block.

The C3K2 uses C3K block to process the information. It has 2 Conv block at start and end following with a series of C3K block and lastly concatinating the Conv Block output and the last C3K block output and ends with a final Conv Block.This block focuses on maintaining a balance between speed and accuracy, leveraging the CSP structure.

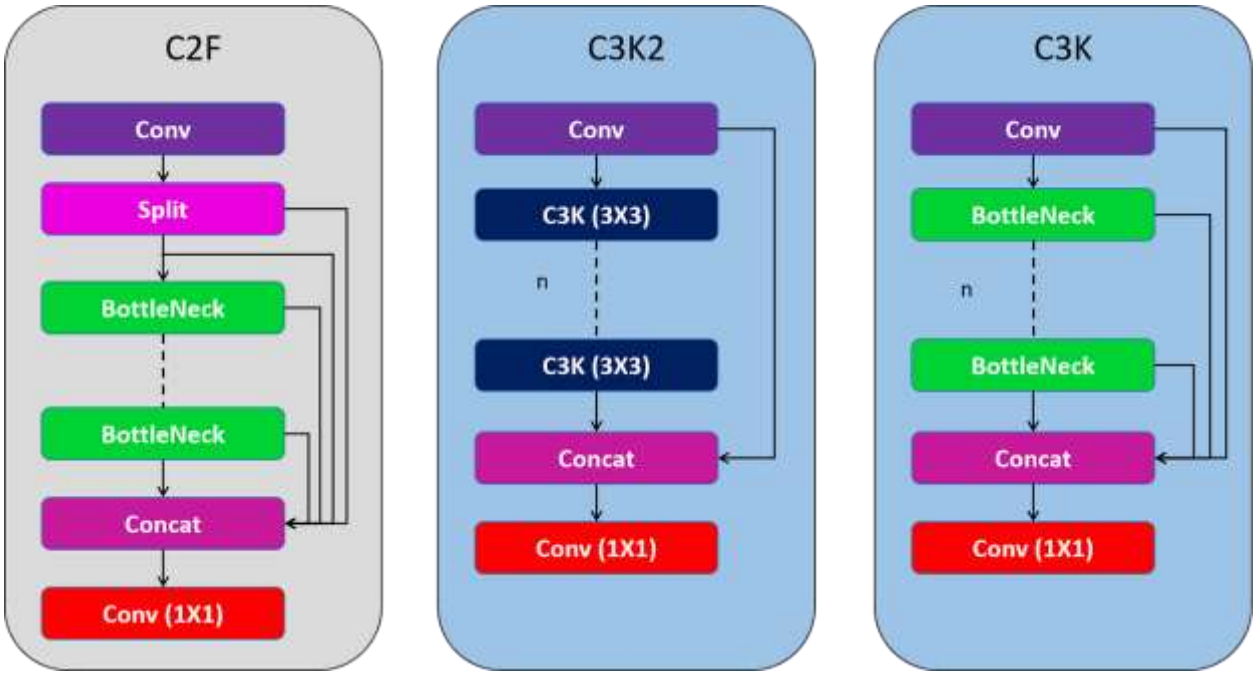

Figure 2.12 Spatial Pyramid Pooling Fast [9]

2.3.3.2 Neck: Spatial Pyramid Pooling Fast (SPFF) and Upsampling

YOLOv11 retains the SPFF module (Spatial Pyramid Pooling Fast), which was designed to pool features from different regions of an image at varying scales. This improves the network's ability to capture objects of different sizes, especially small objects, which has been a challenge for earlier YOLO versions.

SPFF pools features using multiple max-pooling operations (with varying kernel sizes) to aggregate multi-scale contextual information. This module ensures that even small objects are recognized by the model, as it effectively combines information across different resolutions. The inclusion of SPFF ensures that YOLOv11 can maintain real-time speed while enhancing its ability to detect objects across multiple scales.

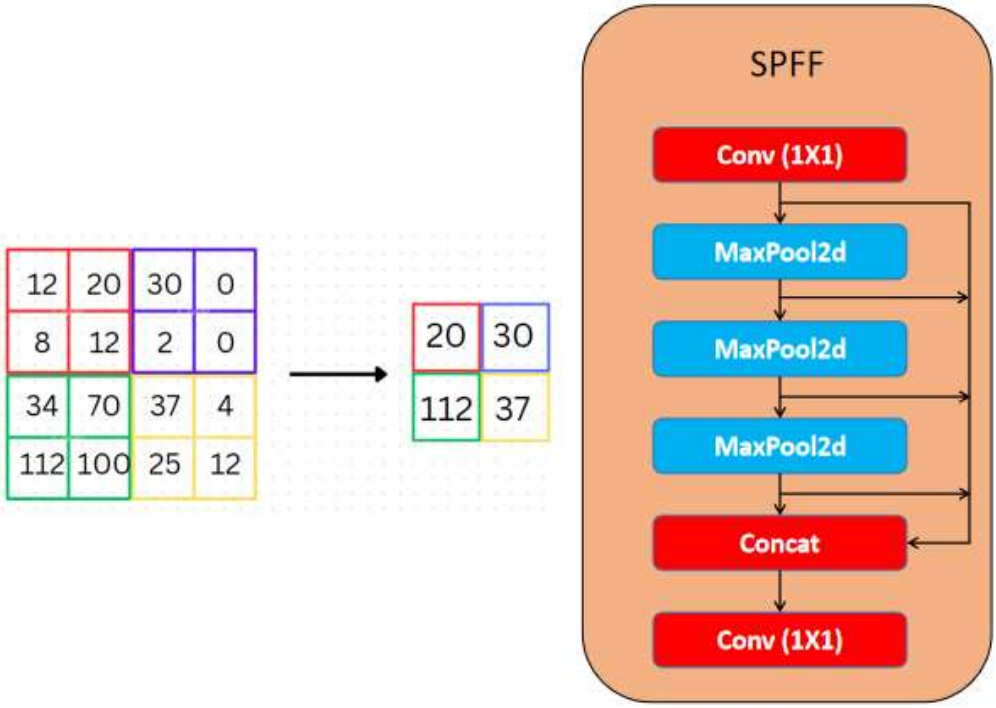

Figure 3. Spatial Pyramid Pooling Fast [9]

2.3.3.3 Attention Mechanisms: C2PSA Block

One of the significant innovations in YOLOv11 is the addition of the C2PSA block (Cross Stage Partial with Spatial Attention). This block introduces attention mechanisms that improve the model's focus on important regions within an image, such as smaller or partially occluded objects, by emphasizing spatial relevance in the feature maps.

a. Position-Sensitive Attention

This class encapsulates the functionality for applying position-sensitive attention and feed-forward networks to input tensors, enhancing feature extraction and processing capabilities. This layer includes processing the input layer with Attention layer and concatinating the input and attention layer output, then it is passed through a Feed forward Neural Networks following with Conv Block and then Conv Block without activation and then concatinating the Conv Block output and the first contact layer output.

b. C2PSA

The C2PSA block uses two PSA (Partial Spatial Attention) modules, which operate on separate branches of the feature map and are later concatenated, similar to the C2F block structure. This setup ensures the model focuses on spatial information while maintaining a balance between computational cost and detection accuracy. The C2PSA block refines the model's ability to selectively focus on regions of interest by applying spatial attention over the extracted features. This allows YOLOv11 to outperform previous versions like YOLOv8 in scenarios where fine object details are necessary for accurate detection.

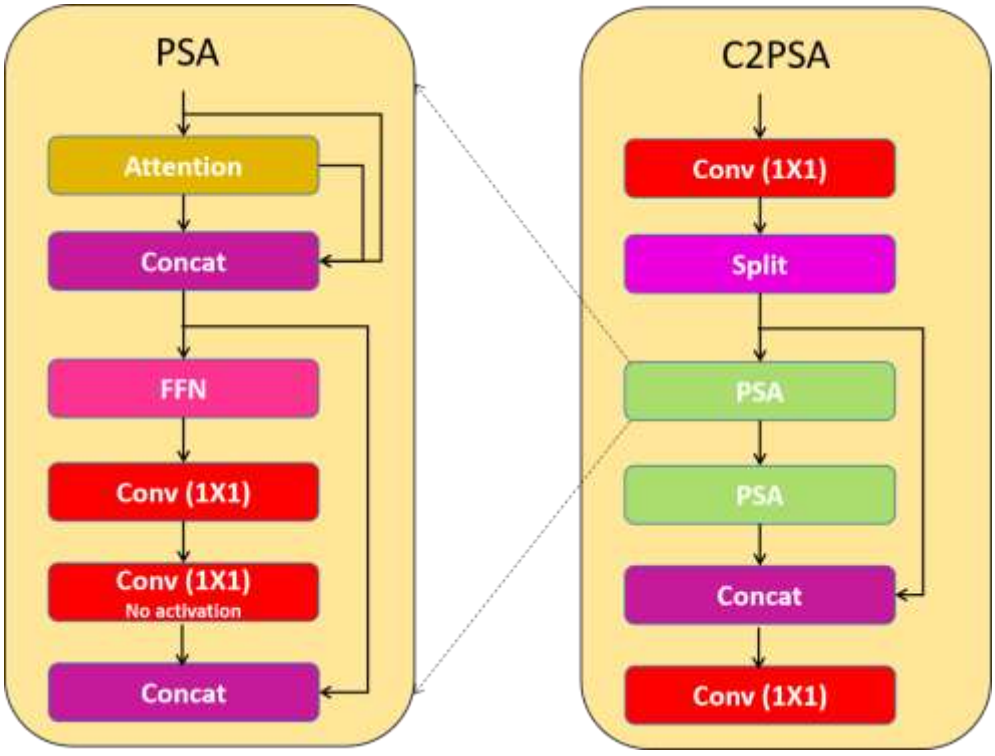

Figure 2.14 C2-Position Sensitive Attention Block (C2PSA) [9]

2.3.3.4 Head: Detection and Multi-Scale Predictions

Similar to earlier YOLO versions, YOLOv11 uses a multi-scale prediction head to detect objects at different sizes. The head outputs detection boxes for three different scales (low, medium, high) using the feature maps generated by the backbone and neck.

The detection head outputs predictions from three feature maps (usually from P3, P4, and P5), corresponding to different levels of granularity in the image. This approach ensures that small objects are detected in finer detail (P3) while larger objects are captured by higher-level features (P5). [9]

2.4 Gradio and Application Deployment

2.4.1 Introduction to Gradio Framework

2.4.1.1 Background and Motivation

In modern machine learning workflows, particularly those involving collaboration between AI engineers and domain experts (such as quality control staff in manufacturing), accessibility represents a critical challenge. Deep learning models are typically developed in controlled research environments and require substantial technical expertise to operate effectively. This technical barrier creates significant obstacles when deploying these models in real-world industrial settings where the end-users may lack programming skills or deep understanding of machine learning concepts.

The gap between model development and practical deployment is particularly pronounced in industrial quality control scenarios, where:

- Technical expertise mismatch: Quality control staff possess deep domain knowledge about defects and manufacturing processes but may lack programming skills
- Real-time decision requirements: Industrial environments demand immediate feedback and decisions
- Trust and interpretability needs: Operators need to understand and trust model predictions before making critical quality decisions
- Continuous improvement cycles: Models need regular updates based on real-world feedback and new defect patterns

To address these challenges, this project adopts Gradio, an open-source Python library designed to democratize access to machine learning models through intuitive web interfaces.

2.4.1.2 Gradio

According to, Gradio was developed as a comprehensive solution to support rapid prototyping, testing, and sharing of machine learning models across interdisciplinary teams [9]. The framework emerged from extensive research involving interviews with 12 machine learning researchers engaged in collaborative projects with domain experts across various fields.

Core philosophy: Gradio is designed to improve accessibility to machine learning models by providing user-friendly interfaces. This characteristic supports industrial deployment where non-technical users are involved.

Primary objectives:

1. Democratize ML access: Enable non-technical users to interact with sophisticated ML models without requiring programming knowledge
2. Facilitate feedback collection: Provide mechanisms for users to flag incorrect outputs, contributing to model improvement cycles
3. Enhance collaboration: Create seamless communication channels between ML developers and domain experts
4. Accelerate deployment: Reduce the time and complexity involved in moving from model development to real-world application

Technical architecture and capabilities

Supported data types: Gradio provides comprehensive support for various data modalities commonly encountered in industrial applications:

- Image Processing: JPEG, PNG, and other standard formats with built-in manipulation tools
- Audio Analysis: WAV, MP3 for acoustic defect detection applications
- Text Processing: Natural language inputs for reports and annotations
- Tabular Data: CSV and structured data for statistical analysis
- Video Streams: Real-time video processing for continuous monitoring

Framework Integration: The library demonstrates exceptional flexibility by supporting major machine learning frameworks:

- PyTorch: Native support for dynamic computational graphs
- TensorFlow/Keras: Comprehensive integration with Google's ML ecosystem
- Scikit-learn: Support for traditional machine learning algorithms
- Custom Functions: Ability to wrap arbitrary Python functions as models

2.4.2 Gradio's Role in Industrial Deployment

2.4.2.1 Addressing Industrial Challenges

The deployment of AI systems in manufacturing environments presents unique challenges that Gradio specifically addresses:

Challenge 1: Technical skill gap

- Problem: Quality control operators possess deep domain expertise but limited programming skills
- Gradio Solution: Web-based interfaces eliminate the need for technical implementation knowledge
- Impact: Enables immediate adoption without extensive training programs

Challenge 2: Real-time decision making

- Problem: Manufacturing processes require immediate feedback on quality assessments
- Gradio Solution: Near-instantaneous model inference with visual feedback
- Impact: Maintains production line efficiency while improving quality control

Challenge 3: Trust and interpretability

- Problem: Black-box models create hesitation among operators to trust automated decisions
- Gradio Solution: Visual interfaces showing detection results, confidence scores, and allowing input manipulation
- Impact: Builds operator confidence through transparent, interpretable results

Challenge 4: Continuous improvement

- Problem: Models need regular updates based on real-world performance
- Gradio Solution: Built-in flagging system for collecting feedback on model performance

- Impact: Creates systematic improvement cycles based on operational data

2.4.2.2 Industrial Deployment Benefits

The implementation of AI-powered defect detection systems in industrial manufacturing environments offers significant operational advantages that address both technical and practical challenges. Modern manufacturing facilities require solutions that can seamlessly integrate into existing workflows while providing real-time feedback to operators and quality control teams. The following table outlines the key industrial deployment benefits of such systems, highlighting how web-based interfaces eliminate the need for complex software installations, while real-time response capabilities ensure immediate defect identification during production processes. These features are particularly valuable for non-technical factory staff who need intuitive tools for quality inspection, and the systematic feedback collection mechanisms enable continuous improvement of the detection models through practical field data.

Table 2.2 Industrial deployment benefits

Feature	**Benefit in industrial context**	**Implementation details**
Web-based interface	No software installation required; runs in any modern browser	Accessible from any device with internet connectivity, including mobile devices for field inspections
Real-time response	Operators receive instant defect feedback from the model	Typical inference time: 50-200ms for YOLOv11 on GPU-enabled systems
Accessible for Non-engineers	Factory staff can use the tool without needing AI or coding skills	Intuitive drag-and-drop interface with clear visual feedback
Feedback collection	Users can "flag" wrong detections, improving future retraining	Systematic collection of misclassified examples for model improvement
Integration with notebooks	Can be embedded in Colab or Jupyter for rapid internal testing	Facilitates rapid prototyping and testing during development phases

Batch processing	Support for multiple image analysis	Enables quality control of entire production batches
Multi-user support	Multiple operators can access the same model simultaneously	Supports distributed quality control across multiple inspection stations

2.4.2.3 Specific Applications in Metal Defect Detection

For metal surface inspection, Gradio provides several specialized capabilities:

Visual Defect Analysis:

- Bounding Box Visualization: Clear marking of detected defects with confidence scores
- Defect Classification: Color-coded labels for different defect types (scratches, dents, corrosion, etc.)
- Severity Assessment: Visual indicators of defect severity levels

Interactive Testing:

- Image Manipulation: Operators can crop, rotate, or adjust brightness to test model robustness
- Comparative Analysis: Side-by-side comparison of original and processed images
- Threshold Adjustment: Real-time adjustment of detection sensitivity

Quality Assurance Integration:

- Report Generation: Automatic generation of inspection reports with timestamps
- Data Logging: Systematic recording of all inspections for traceability
- Alert Systems: Automatic notifications for critical defects

CHAPTER 3: CURRENT SITUATION ANALYSIS

3.1. Manual Inspection Process and Its Limitations at Quoc Quang Electro-Acoustic Co., Ltd

At Quoc Quang Electro-Acoustic Co., Ltd, a company specializing in speaker production, the inspection of incoming raw materials plays a critical role in ensuring product quality. However, the current inspection process is predominantly manual, which poses significant challenges in terms of efficiency, consistency and accuracy.

The existing manual inspection workflow consists of four main steps as illustrated in Figure 15:

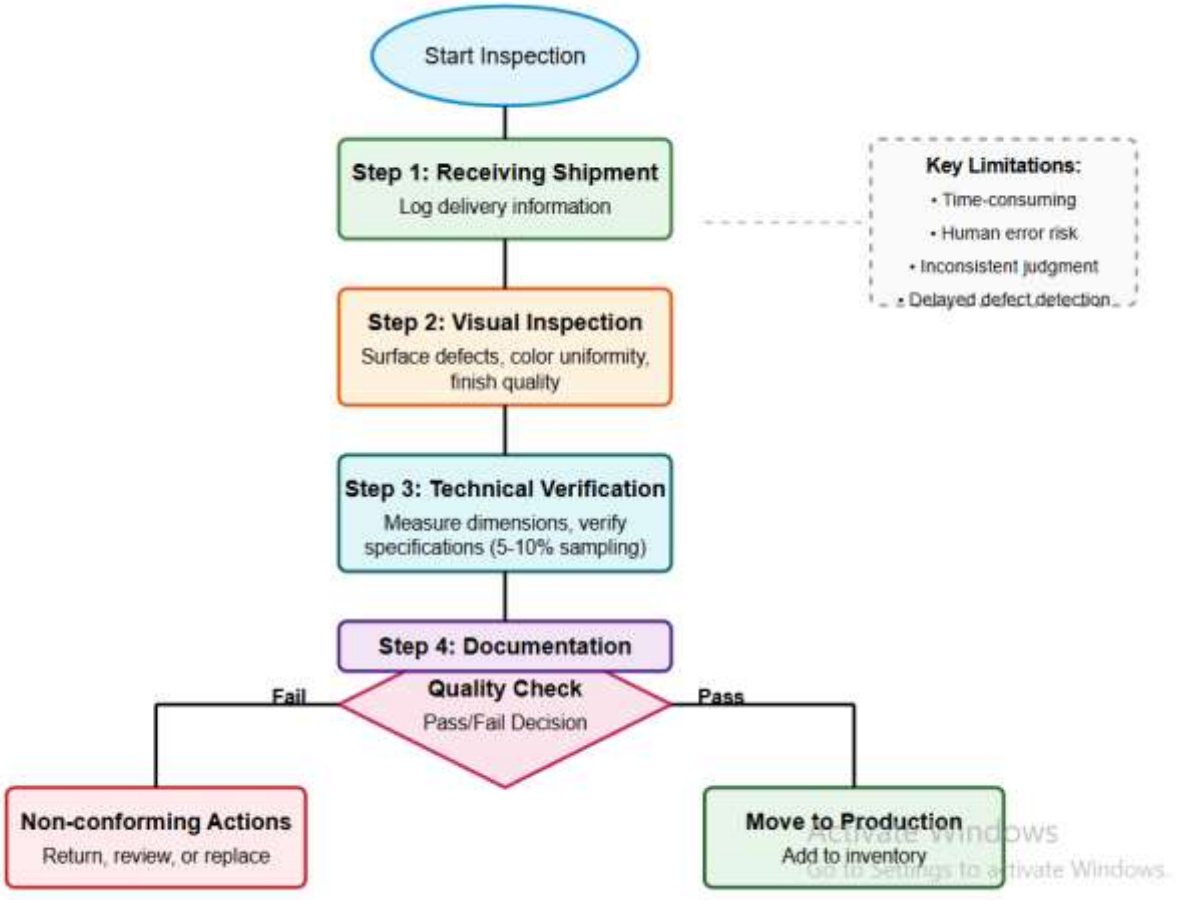

Figure 43.1 Current manual inspection process at Quoc Quang Electro-Acoustic Co., Ltd

3.1.1 Current Inspection Workflow

Step 1: Receiving the Shipment

Upon arrival of raw materials from suppliers, the warehouse team performs the following actions:

- Temporarily store materials in a designated storage area.
- Checks and logs the delivery information:
    - Supplier name
    - Type and quantity of materials
    - Batch or lot number
    - Delivery date and time
    - Condition of packaging

Once this is completed, the Quality Control (QC) department is notified to begin quality inspection of the materials.

Step 2: Visual Inspection

The QC personnel perform a detailed visual assessment of the materials to detect surface-level defects and evaluate overall appearance. The key focus areas include:

- Surface defects:
    - Inspecting common visible issues such as crack, pitted, patches
    - These are typically identified through visual observation under appropriate lighting conditions.
- Color uniformity:
    - Ensuring consistency in color and coating across the entire part.
    - This is especially important for visible parts like speaker grills and aluminum frames, where aesthetic consistency is critical.

- Finish quality:
    - Checking for burrs or sharp edges, especially at cutting areas.
    - Ensuring corners are properly rounded and not hazardous.
    - Verifying hole positioning and alignment according to design specifications.

In some cases, the QC team uses simple measuring instruments such as calipers, rulers, and micrometers to assist in validating visual observations.

Step 3: Technical Specification Verification

After visual inspection, the QC team randomly selects 5–10% of the batch for a more in-depth technical inspection. These samples are compared against official technical drawings or internal company standards.

Specific inspection criteria include:

- Hole diameter and spacing:
    - Measured using digital calipers to confirm alignment and adherence to technical tolerances.
- Frame thickness:
    - Checked at multiple points to verify uniformity and structural integrity.
- Material type and surface coating:
    - Confirmed via supplier documentation (material certificates).
    - Physical checks are also performed for coating evenness, absence of peeling, and bubble-free surfaces.

Step 4: Documentation and Quality Decision

Following inspection, all observations are compiled in the Incoming Material Inspection Report (IMIR), which includes:

- Material details (type, lot number, quantity)
- Inspection results (pass/fail, visual notes, measurements)
- Inspector’s name and date

Based on the results:

- Approved batches are moved into inventory for production use.
- Non-conforming batches are:
    - Returned to the supplier, or
    - Held for further review, or
    - Replaced with a new delivery

This documentation ensures traceability and supports consistent quality management across the production chain.

3.1.2 Limitations of the Current Inspection Process

Despite the structured approach, this manual inspection process suffers from multiple drawbacks:

1. Time-consuming:
    - On average, each part takes 2–3 minutes to inspect.
    - With thousands of units inspected daily, the total QC time can exceed 100 hours per week.
2. High risk of human error:
    - The effectiveness of the inspection heavily depends on the experience and alertness of individual inspectors.
    - Subtle defects, such as micro-scratches < 0.2 mm width or internal inclusions, are often overlooked due to human visual thresholds. Based on internal logs, approx. 7–10% of these defects go undetected.
3. Inconsistency:
    - Quality judgment varies between inspectors, leading to inconsistent assessments. For instance, inspector disagreement on "acceptable" vs. "rework" classification occurs in 12% of borderline cases, resulting in production delays or mixed quality.
    - This leads to subjective decisions and variation in accepted quality levels.
4. Delayed defect detection:
    - In some cases, defects are not discovered until much later in the production line, such as during:
        - Speaker assembly (crack, misalignment)
        - Paint coating (pitting, coating failure)
        - Final QC (metal patches, deformation)
    - This causes production delays, scraping of components, or increased rework costs.

These limitations highlight the need for a more reliable, consistent, and efficient inspection system that can overcome the inherent challenges of human-dependent quality control processes.

3.2 Examples of Missed Defects and Their Consequences

Inadequate detection of defects during the manual inspection process has led to significant negative impacts on production efficiency, product quality, and customer satisfaction. The company's internal analysis has revealed several critical defect categories that are frequently missed during initial inspection, causing problems at later production stages.

3.2.1 Common defect types missed during inspection

The most common types of defects that typically go undetected during manual inspection include:

- Crack: Fine, spider-web-like cracks appearing on the surface of metal due to stress or temperature variation, often seen as faint, irregular lines. Linear marks or abrasions on the surface, usually caused by physical contact with hard or sharp objects during storage or handling.
- Pitted: Small, round indentations or holes caused by corrosion or impact, which compromise the uniformity of the surface.
- Patches: Uneven or abnormal surface textures appear as blotchy areas, often due to improper coating or surface contamination.

Each defect type presents unique challenges in terms of detection and impact. The tables below provide quantitative data on when these defects are typically detected in the production process and their consequential impact on operations and product quality.

Table 4 illustrates the common defects that are typically detected too late in the production process, along with their causes and impacts:

Table 3.1 Common defects detected too late due to input material errors

Defect Type	Detected During Process	Cause	Impact
Crack	Speaker Assembly	Poor storage or handling	Affects appearance; lowers product value
Pitting	Paint Coating	Rust on raw material surface	Paint adhesion failure; cosmetic degradation

	Final Quality Control (QC)	Contaminants in raw metal	Entire product rejected at final stage
Patches	Material Preparation	Improper material composition or finish	Paint defects; inconsistent surface texture; poor aesthetics

Further analysis of these missed defects reveals the quantitative impact on production and quality metrics as shown in Table 5:

Table 3.2 Missed defects and their impact

Defect Type	Average Miss Rate	Estimated Loss per Case
Crack	8%	15,000 VND per unit (scrapping required)
Patches	6%	8% increase in customer returns
Pitted	5%	Reduces product lifespan by 20%

These defects have had substantial business impacts for Quoc Quang Electro-Acoustic Co., Ltd. Missed crack defects have resulted in 15% product devaluation on export orders. Additionally, customer complaints related to pitting and patch defects increased by 10% in Q4/2024.

3.2.2 Quantitative Impact Analysis

The missed defects have created a cascading effect throughout the production chain:

1. Increased rework: approximately 12% of products require rework due to defects identified at later stages, resulting in an estimated additional labor cost of 45 million vnd monthly.
2. Production delays: late-stage defect detection has increased production cycle time by an average of 14%, reducing overall manufacturing capacity.
3. Warranty claims: the company has seen a 10% increase in warranty claims over the past six months directly attributable to defects that originated from inadequately inspected raw materials.
4. Quality control overhead: the need for more rigorous final inspections has increased qc staffing requirements by 15%, creating additional operational expenses.

These findings clearly demonstrate that the current manual inspection process is insufficient for maintaining the desired quality standards and operational efficiency at quoc quang electro-acoustic co., ltd.

3.3 Motivation for an Automated Web-Based Inspection System

The detailed analysis of the current manual inspection process at Quoc Quang Electro-Acoustic Co., Ltd has revealed significant limitations that directly impact product quality, operational efficiency, and business profitability. These findings create a compelling case for implementing an automated inspection solution to address these challenges.

3.3.1 Key Drivers for Automation

The primary motivations for transitioning from manual to automated inspection include:

1. **Process efficiency enhancement**
    - The current manual inspection process requires 2-3 minutes per part, resulting in over 100 hours of inspection time weekly.
    - An automated system could potentially reduce inspection time from minutes to seconds per part, enabling significantly higher throughput.
    - This time efficiency would allow the quality control team to focus on more complex quality issues rather than routine inspections.
2. **Error reduction and consistency**
    - The documented 7-10% miss rate for subtle defects represents significant quality risks.
    - Quality judgment inconsistency between inspectors (12% disagreement rate) leads to unpredictable outcomes and quality variations.
    - An AI-powered system would apply consistent evaluation criteria across all inspections, eliminating human variability.
3. **Early defect detection**
    - The quantified business impact of late defect detection (15% product devaluation on exports, 10% increase in customer complaints) demonstrates the critical need for earlier and more accurate defect identification.
    - Automated detection at the material receipt stage would prevent defective components from entering the production stream entirely.
4. **Enhanced traceability and quality management**
    - The digital nature of automated inspection would enable comprehensive data collection and analysis.
    - Each inspection result could be automatically logged, creating an auditable trail of quality assurance activities.
    - Historical data would facilitate continuous improvement of both the inspection system and upstream supplier quality.

3.3.2 Proposed Solution Approach

To address these challenges, this project proposes the development of a web-based automated defect detection system leveraging deep learning techniques. The system will provide the following key capabilities:

1. **User-friendly web interface**
    - Enable users to upload images of raw materials through an intuitive web platform.
    - Eliminate the need for specialized equipment or technical expertise to perform inspections.
2. **Advanced defect detection**
    - Implement a two-stage deep learning pipeline utilizing YOLOv11 for defect region detection and precise defect classification.
    - Detect subtle defects that may be missed by the human eye, including micro-crack, minor pitted, and early-stage pitting.
3. **Real-time feedback**
    - Provide immediate analysis results indicating the location, type, and severity of detected defects.
    - Flag defective materials instantly at the receiving stage, preventing them from proceeding to production.
4. **Enhanced traceability**
    - Store all inspection images with timestamp and defect annotations in a searchable database.
    - Generate comprehensive inspection reports for quality audits and process optimization.

This solution directly addresses the identified limitations of the current manual process while providing a scalable framework for future enhancements. The following chapters will detail the methodology, implementation, and evaluation of this automated inspection system.

The comprehensive analysis of the current manual inspection process has clearly demonstrated the need for a more sophisticated, automated approach to quality control at Quoc Quang Electro-Acoustic Co., Ltd. The limitations of human inspection—including time

constraints, inconsistency, and error rates—have resulted in quantifiable negative impacts on product quality and business performance.

By implementing the proposed web-based automated inspection system, the company can expect substantial improvements in inspection accuracy, efficiency, and consistency. This technological advancement represents not just an operational enhancement but a strategic investment in long-term quality assurance and customer satisfaction.

The next chapters will explore in detail how this solution will be designed, implemented, and validated to address the specific challenges identified in the current inspection process.

CHAPTER 4: METHODOLOGY

4.1 Research Methodology

4.1.1 Research Methodology Framework

This study uses a systematic, evidence-based research methodology based on established deep learning practices for object detection applications:

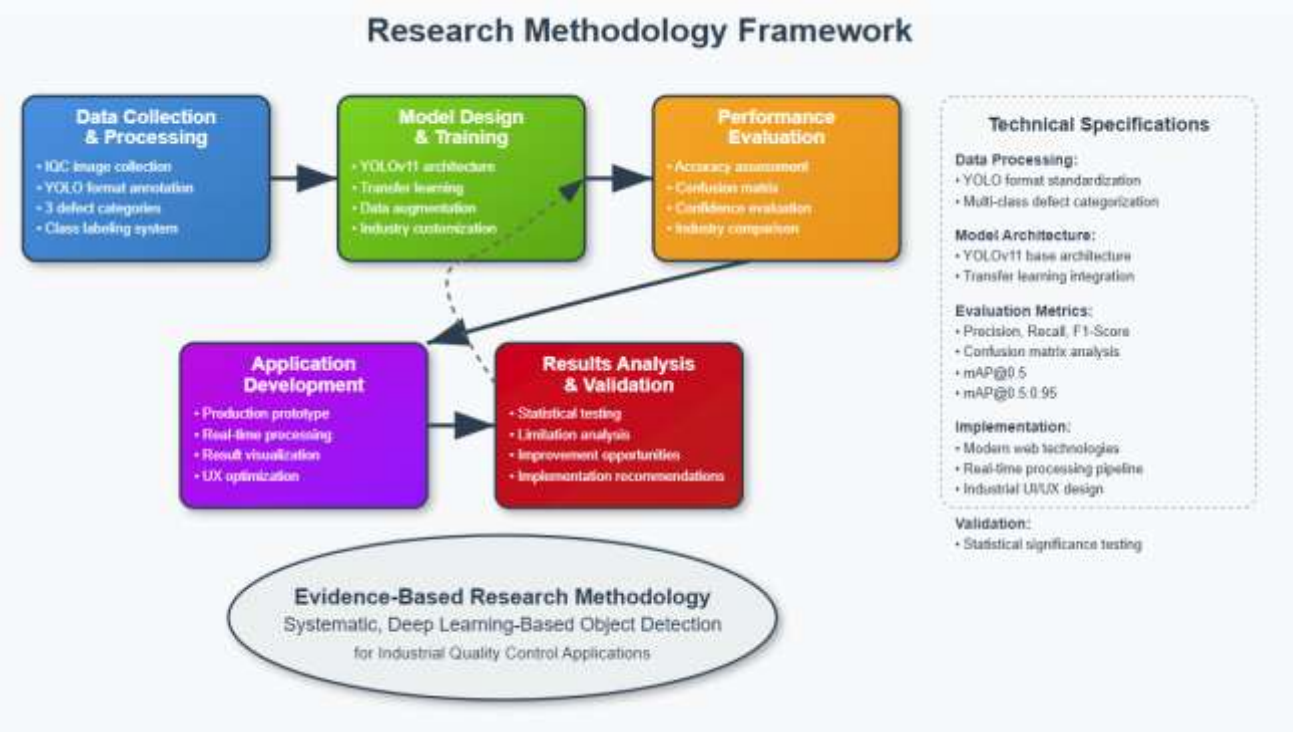

Figure 5. Research methodology framework

4.1.1.1 Data Collection and Processing

- Systematic collection of defective component images from operational IQC processes
- Implementation of standardized annotation methods using YOLO format specifications
- Development of comprehensive class labeling system for three distinct surface defect categories

4.1.1.2 Model Design and Training

- Implementation of YOLOv11 architecture with industry-specific customizations
- Application of advanced transfer learning methods
- Integration of sophisticated data augmentation strategies to improve model generalization capabilities

4.1.1.3 Performance Evaluation Framework

- Comprehensive accuracy assessment using standardized object detection metrics
- Statistical validation through confusion matrix analysis and confidence level evaluation
- Comparative performance testing against industry standards

4.1.1.4 Application Development and Implementation

- Development of production-ready prototype interface using modern web technologies
- Implementation of real-time processing capabilities with comprehensive result visualization
- Integration of user experience optimization principles for industrial application contexts

4.1.1.5 Results Analysis and Validation

- Systematic interpretation of experimental results with statistical significance testing
- Identification of model limitations and improvement opportunities
- Development of evidence-based recommendations for industrial implementation

4.1.2 Research Design and Approach

The research adopts a quantitative experimental methodology based on supervised machine learning principles. The approach follows a structured development lifecycle encompassing data preparation, model selection, training procedures, and application development phases. The experimental design employs YOLOv11 deep learning architecture for object detection tasks, specifically adapted for industrial defect identification applications.

The research methodology addresses three critical defect types commonly encountered in manufacturing processes: crack, pitted, patches.

Each image in the classification datasets belongs to one of the three predefined defect classes. A brief description of each defect type is provided below:

Table 4.1 A brief of each defect type

Defect Type	Description
Crack	Fine, spider-web-like cracks appearing on the surface of metal due to stress or temperature variation, often seen as faint, irregular lines.

	Linear marks or abrasions on the surface, usually caused by physical contact with hard or sharp objects during storage or handling.
Pitted	Small, round indentations or holes caused by corrosion or impact, which compromise the uniformity of the surface.
Patches	Uneven or abnormal surface textures appear as blotchy areas, often due to improper coating or surface contamination.

Figure 6. Crack

Figure 4.3 Crack

Figure 7. Pitted

Figure 4.5 Pitted

Figure 4.6 Patches

4.1.3 Dataset Development Methodology

The dataset development process adheres to established computer vision practices, focusing on real-world applicability. The primary dataset consists of 871 defect images collected from GGEC Vietnam's actual production lines and raw material inspection stages, ensuring high relevance to real manufacturing conditions. The data acquisition phase spans multiple production cycles and encompasses diverse metal components to capture a wide variety of defect patterns and surface conditions.

Annotations follow the YOLO format, with each defect instance labeled using precise bounding box coordinates and its corresponding class. The annotation process is carefully

reviewed to maintain consistency and accuracy, including manual verification of edge cases and visual anomalies by multiple annotators.

The dataset is partitioned into two primary subsets:

Training set: 763 images

Validation set: 108 images

This split strategy, equivalent to approximately 87% training and 13% validation, ensures the model has sufficient data for learning while providing an independent set for hyperparameter tuning and performance monitoring. A separate test set is not defined in the traditional sense; instead, post-training evaluations are performed on practical deployment images collected from real production samples not seen during training or validation. The lack of an independent test set is a study limitation. Validation set results may not fully reflect real-world generalization. Future work should expand datasets to include dedicated test splits. This approach reflects the intended application context of the system and allows assessment under realistic operating conditions.

To maintain fairness during training, class balance across the two subsets is preserved, preventing model bias toward dominant defect categories. This partitioning methodology aligns with the project's focus on industrial deployment and practical performance evaluation rather than strict academic benchmarking.

4.1.4 YOLOv11 Model Selection and Configuration

The model selection methodology evaluates YOLOv11 architecture based on specific requirements for industrial defect detection applications. YOLOv11 offers advantages including real-time processing capabilities, single-stage detection efficiency, and scalable architecture variants to accommodate different computational constraints. The selection criteria prioritize detection accuracy, inference speed, and deployment flexibility for industrial environments.

YOLOv11 architecture comprises three fundamental components: the backbone network utilizing CSPDarknet53 for feature extraction, the neck network implementing PANet structure for multi-scale feature fusion, and the head network providing detection outputs including bounding box coordinates, confidence scores, and class probabilities. The configuration methodology considers optimal balance between model complexity and detection performance for the specific defect detection requirements.

The methodology includes evaluation of different YOLOv11 variants (YOLOv11s, YOLOv11m, YOLOv11l, YOLOv11x) to determine the most suitable architecture for the application requirements. Model selection criteria encompass accuracy metrics, computational efficiency, memory requirements, and deployment considerations for practical industrial implementation.

4.1.5 Training Methodology and Procedures

The model training process adopts a transfer learning strategy using pretrained weights from the COCO dataset as initialization for YOLOv11. This approach accelerates convergence and improves detection performance on limited industrial defect data.

Training is performed in two main phases:

1. Initial training phase: The model is trained from COCO weights using the full dataset with default hyperparameters for 200 epochs, applying early stopping with a patience of 50.
2. Fine-tuning phase: The best checkpoint from phase one is further fine-tuned with optimized parameters, including adjusted confidence threshold, image size (imgsz=640), and learning rate for 250 epochs, applying early stopping with a patience of 50.

To improve model generalization, various data augmentation techniques are applied, including rotation, scaling, brightness adjustments, and mosaic augmentation. These augmentations simulate diverse defect appearances while preserving real-world characteristics.

The training configuration uses an adaptive learning rate schedule, batch sizes suited for Google Colab's GPU environment, and standardized input dimensions. Early stopping is used to prevent overfitting and monitor performance on the validation set.

All training is conducted on Google Colab, which provides accessible GPU resources and ensures reproducibility. Resource monitoring and checkpoint saving are employed to manage computational efficiency and preserve best-performing models.

4.1.6 Evaluation Methodology and Metrics

The performance of the YOLOv11 object detection model is evaluated using a set of standard metrics widely adopted in computer vision tasks. These metrics provide a quantitative measure of the model's ability to detect and localize defects on metal surfaces accurately and efficiently. These metrics include Precision, Recall, Intersection over Union (IoU), Mean

Average Precision (mAP), and the F1 Score [10]**.** Precision is the proportion of true positives (TP) to the sum of true positives and false positives (FP), and it measures how many of the predicted positive instances are actually correct. It is given by the Formula (1):

$$Precision = \frac{TP}{TP+FP} \text{ (1)}$$

Where:

- TP (True Positive): Correctly predicted bounding boxes that match ground-truth defects.
- FP (False Positive): Incorrectly predicted bounding boxes that do not correspond to any ground-truth defect.

Recall, on the other hand, measures the proportion of true positives (TP) to the sum of true positives and false negatives (FN), and it assesses how many of the actual positive instances are correctly identified by the model. It is calculated as Formula (2):

$$Recall = \frac{TP}{TP+FN} \text{ (2)}$$

Where:

- FN (False Negative): Ground-truth defects that were missed by the model.

To provide a balance between precision and recall, the F1 Score is used. The F1 Score is the harmonic mean of precision and recall, and is calculated by the Formula (3):

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \text{ (3)}$$

Intersection over Union (IoU) quantifies the overlap between the predicted bounding box and the ground-truth box. IoU is given by the Formula (4):

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \text{ (4)}$$

Mean Average Precision is the primary evaluation metric for object detection. It is computed by taking the average of the precision values across different recall levels for each class and then averaging across all classes, as shown below Formula (5):

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \text{ (5)}$$

The mAP at various IoU thresholds, such as mAP@50 and mAP@50-95, is commonly used to evaluate object detection models. These metrics—Precision, Recall, IoU, mAP, and F1 Score—are used together to assess the overall performance of the defect detection system. Precision and recall provide insights into the model's ability to minimize false detections and missed defects, while IoU and mAP quantify how accurately the predicted bounding boxes match the actual defect locations. The F1 Score offers a balanced view by combining both precision and recall into a single metric, supporting a more comprehensive evaluation of the model's detection capabilities.

4.2 System Design and Application Development

4.2.1 Application Development Methodology

The application development methodology employs Gradio framework for creating an interactive web-based interface integrated with the trained YOLOv11 model. Gradio provides seamless integration capabilities with machine learning models while offering professional interface development with minimal coding requirements. The development approach prioritizes user experience, functionality, and deployment accessibility.

The application architecture follows modular design principles, separating user interface components from model inference procedures to ensure maintainability and scalability. The methodology includes comprehensive workflow design from image input through preprocessing, model inference, post-processing, and result visualization stages. Error handling and user guidance procedures ensure robust application performance across various usage scenarios.

Integration methodology addresses seamless connection between the trained model and the user interface, including proper resource management, session handling, and performance optimization for concurrent user access. The development procedures ensure compatibility with Google Colab environment while maintaining professional application standards suitable for industrial deployment.

4.2.2 Web-based Interface using Gradio

To enhance usability and demonstrate the real-time performance of the trained YOLOv11 model, a web-based application was developed using the Gradio framework. Gradio provides a simple yet powerful interface to interact with machine learning models directly in the browser without the need for complex frontend development.

4.2.2.1 Application Architecture

The application follows a client-server architecture where:

- **Backend**: The trained YOLOv11 model is loaded using Ultralytics API.
- **Frontend**: Gradio provides an upload interface, confidence/IoU sliders, and real-time result rendering.
- **Deployment**: The app runs in a Google Colab environment, allowing users to test directly on custom images.

4.2.2.2 Interface Components

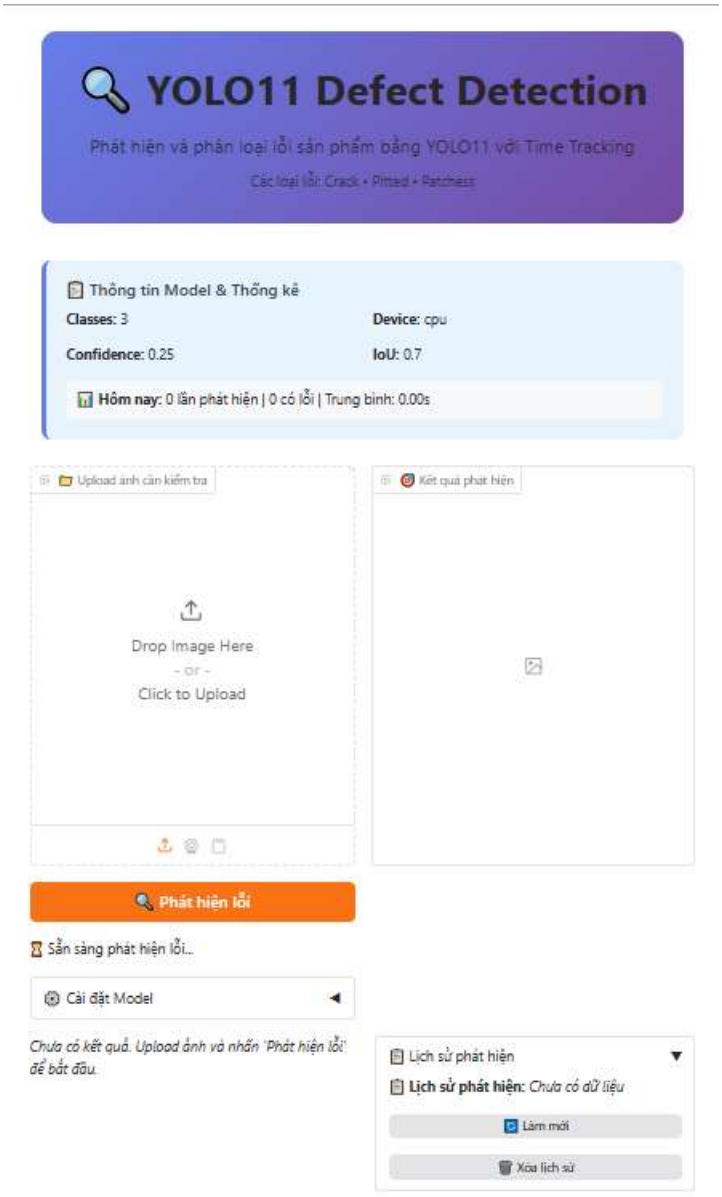

Figure 8. Gradio-based defect detection interface showing the upload component and model information panel.

- **Model Information Panel**: Displays the number of classes, loaded model, current confidence threshold, and device (CPU/GPU).
- **Image Upload**: Users can drag & drop or click to upload a steel surface image.
- **Detection Trigger**: Upon clicking the "Detect Defects" button, the image is analyzed and the output is shown with bounding boxes and class labels.
- **Adjustable Parameters**: Users can change the confidence threshold and IoU threshold to optimize detection results.

4.2.2.3 User Flow

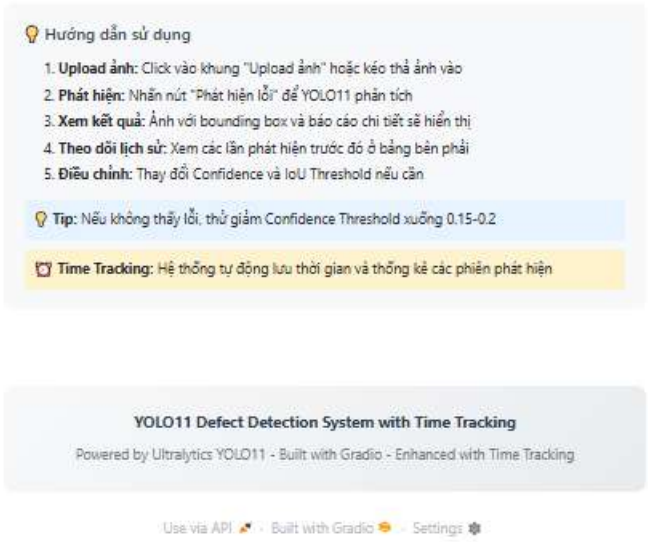

Figure 4.3 Application User Guide

1. Upload an image via the Gradio UI.
2. Set parameters (Confidence, IoU).
3. Click "Detect Defects".
4. The system runs inference using YOLOv11 and displays the result with bounding boxes and defect labels.
5. If no results are shown, the user is advised to lower the confidence threshold.

4.2.2.4. Implementation Notes

- The application runs in a Google Colab notebook using gr.Interface() from the gradio library.
- The predict() function loads the YOLOv11 model from best.pt and returns the image with detections.
- It supports live reloading and interface sharing via public links if needed.

CHAPTER 5: EXPERIMENTAL RESULTS AND ANALYSIS

5.1. Experimental Setup

5.1.1. Experimental Environment

To ensure efficient training and reasonable computation time, all experiments were conducted on Google Colab Pro with the following specifications:

- Platform: Google Colab Pro
- GPU: NVIDIA Tesla T4 (16GB VRAM)
- Framework: YOLOv11 (Ultralytics) – an enhanced version of YOLOv8 with improvements in accuracy and speed
- Programming Language: Python 3.10
- Libraries: PyTorch, Ultralytics, OpenCV, Matplotlib

The dataset used in the experiments was collected and labeled manually based on real production images provided by the company.

- Dataset Type: Custom-built from real-world industrial scenarios
- Number of images: 763 training images, 108 validation images
- Number of defect classes: 3 (crack, pitted, patches)
- Annotation Tool: LabelImg with YOLO format

The evaluation metrics selected to assess model performance are:

- Precision – proportion of correct positive predictions
- Recall – proportion of actual positives correctly identified
- mAP@0.5 – mean Average Precision at IoU threshold 0.5
- mAP@0.5:0.95 – mean Average Precision averaged over IoU thresholds from 0.5 to 0.95 (step 0.05)

5.1.2. Experiment Configuration

The training was configured as follows:

- Input image size: 640 × 640 pixels
- Batch size: 16
- Number of data loading workers: 4
- Training device: CUDA-enabled GPU

- Mixed precision (AMP): Enabled for faster training and lower memory usage

The training process was carried out in two phases:

1. Initial training phase: from scratch for 200 epochs using default hyperparameters.
2. Fine-tuning phase: Continued 250 additional epochs using the best weights from the first phase (best.pt), with optimized hyperparameters for better convergence.

5.2. Optimization Process and Evaluation

All evaluation metrics reported in this section, including Precision, Recall, F1 Score, mAP@0.5, and mAP@0.5:0.95, are computed on the validation set. The validation dataset, consisting of 108 images, is separate from the training set and was not seen by the model during the training process. This ensures an unbiased assessment of the model's ability to generalize to unseen data. All performance curves and confusion matrix results presented in subsections 5.2.1 and 5.2.2 reflect the model's detection accuracy, robustness, and classification consistency when applied to real-world validation samples collected from the manufacturing environment.

5.2.1. Performance of Initial Model (Model 1)

5.2.1.1 Results of model initial training

a. Evaluation metrics and results

Observed results:

Table 5.1 Evaluation metrics and results (model 1)

Metric	Value
Precision	0.678
Recall	0.579
mAP@0.5	0.566
mAP@0.5:0.95	0.257

- mAP@0.5: Reached around 0.5-0.57, showing only moderate detection accuracy.
- mAP@0.5:0.95: Remained low at 0.2–0.28, indicating poor generalization across varying IoU thresholds.
- Precision/Recall: Fluctuated significantly, with averages around 0.6 for both.

Figure 25 illustrates the evolution of precision and recall over training epochs using the YOLOv11 model. Both metrics show an overall upward trend, particularly until around epoch

175. Precision fluctuates more significantly than recall, indicating some inconsistency in the model's ability to consistently produce accurate detections. However, in the later epochs, both precision and recall stabilize around **0.6**, reflecting a balanced ability to correctly detect and sufficiently cover the defect instances. The noticeable fluctuations suggest that the model still lacks full stability.

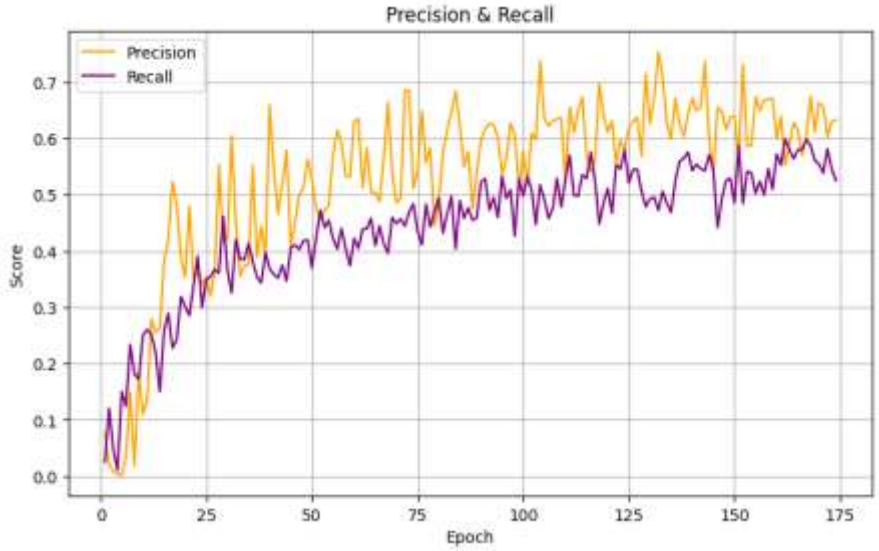

Figure 9 Precision & Recall trends over 175 training epochs (model 1)

The mAP@0.5 metric steadily increases and peaks at approximately **0.6**, indicating moderate accuracy in localizing defects at an IoU threshold of 0.5. In contrast, mAP@0.5:0.95—which represents the model's generalization across multiple IoU thresholds—remains low at around **0.22**, with no significant improvement beyond epoch 125. This implies that the model struggles to maintain detection accuracy when varying the overlap threshold, which reflects a limitation in its ability to generalize to more complex or realistic defect scenarios.

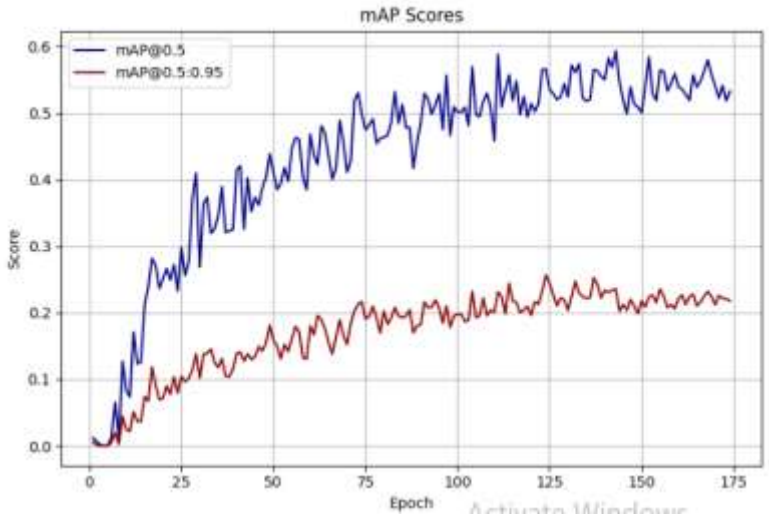

Figure 5.2 mAP Scrores showing model accuracy and generalization (model 1)

The F1 Score curve reflects the harmonic mean of Precision and Recall. Initially low, the score improves significantly during the early training stages and stabilizes around 0.6 after 175 epochs. This indicates that the model achieved a balanced trade-off between precision and recall, ensuring reliable defect detection performance.

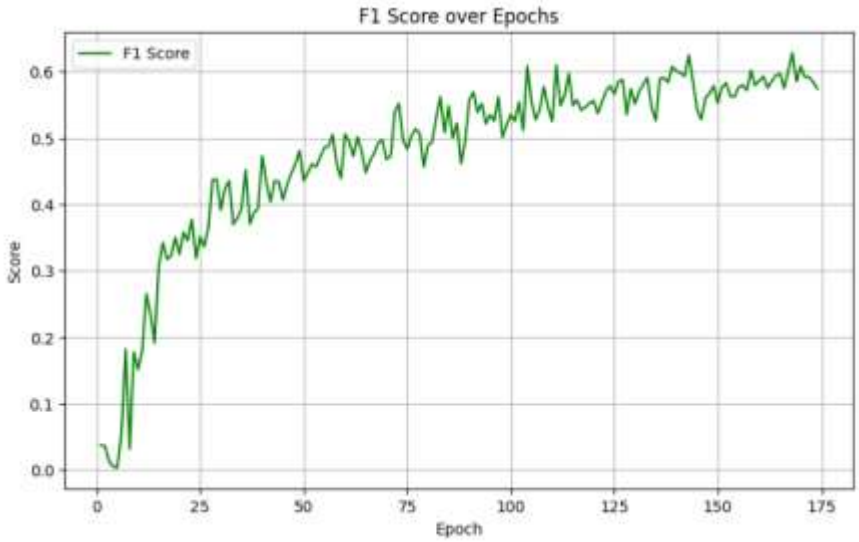

Figure 5.3 F1 Score progression over epochs, highlighting model's convergence toward balanced performance.

b. Confusion matrix analysis

The confusion matrix illustrates the performance of the steel defect detection model across three classes: crack, pitted, and patches. The model shows high accuracy with an overall score of 97.0%. Specifically, it correctly classified 21 out of 22 crack images, all 35 pitted images, and 42 out of 44 patches images. Minor misclassifications occurred, including one crack misidentified as pitted, and two patches misclassified—one as crack and one as pitted. Among the classes, "patches" achieved the highest F1-score (0.977), followed closely by "pitted" (0.972) and "crack" (0.955). These results indicate that the model is highly effective, with only a few classification errors that could be further reduced with more training data or fine-tuning.

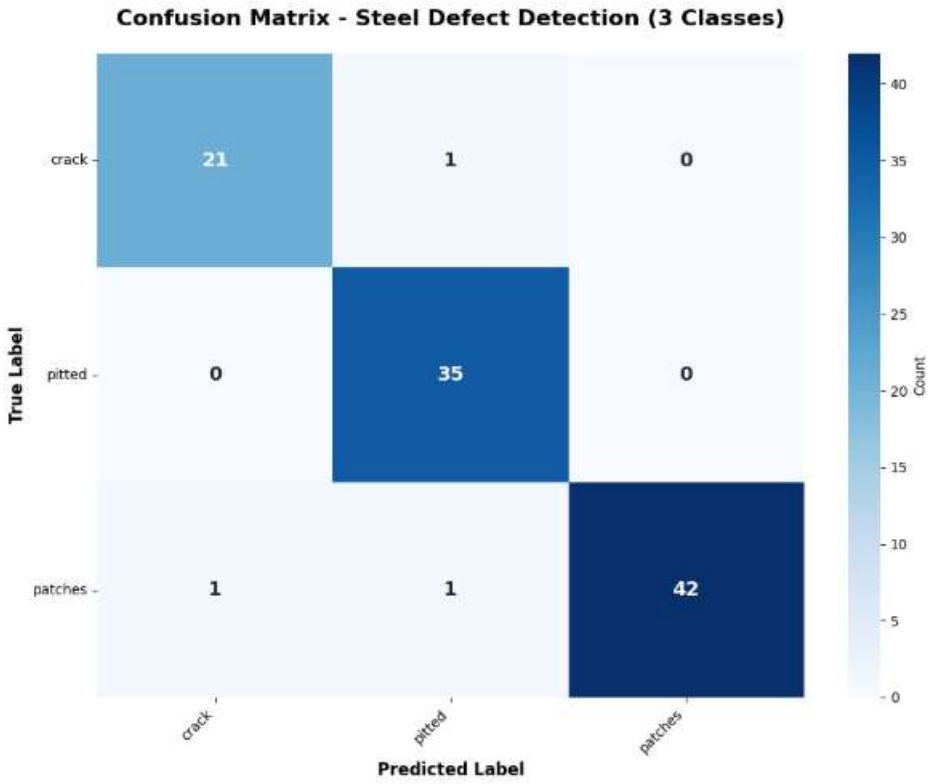

Figure 5.4 Confusion matrix (model 1)

5.2.2. Fine-Tuning with Optimized Settings (Model 2)

To address the limitations of the initial training phase, a fine-tuning strategy was implemented using the best weights from the first training (best.pt) as the base. The objective was to improve convergence, reduce overfitting, and enhance detection accuracy.

Fine-tuning strategy:

Table 5.2 Fine-tuning strategy

Lower Learning Rate:	lr0 = 0.001 # Reduced from 0.005
	lrf = 0.001 # Lower final LR
	warmup_epochs = 3 # Reduced from 5

Mild Data Augmentation (to avoid overfitting):	hsv_h = 0.015 # Lower hue variation
	hsv_s = 0.7 # Lower saturation shift
	hsv_v = 0.4 # Lower brightness shift
	degrees = 10.0 # Smaller rotation
	translate = 0.1 # Lower image translation
	mixup = 0.1 # Less blending between images

Optimized Loss Weights	cls = 1.2 # Increased focus on classification
	box = 7.5 # Strong bounding box regression
	dfl = 1.5 # Balanced distribution focal loss

Early Stopping & Regularization	epochs = 250 patience = 50 weight_decay = 0.0005 # L2 regularization to reduce overfitting

5.2.2.1 Evaluation Metrics and Results

Outcomes of Fine-Tuning:

Table 5.3 Evaluation metrics and results (model 2)

Metric	Value
Precision	0.9
Recall	0.8
mAP@0.5	0.801
mAP@0.5:0.95	0.502

- Improved training performance: Training stopped early at epoch 213 (best at epoch 163) due to no further improvement, indicating convergence.
- mAP@0.5 reached 0.821, and mAP@0.5:0.95 reached 0.502, showing good improvement in both localization and generalization across defect types.
- Precision and Recall became more consistent: Precision peaked at 0.911 (patches), and Recall reached up to 0.82, reflecting better detection capabilities.
- The model shows stronger generalizability on the validation set with reduced variance.

The chart shows the evolution of **Precision (orange)** and **Recall (purple)** over 213 training epochs. Initially, both metrics increased sharply, indicating effective learning during early stages.

- From epoch 0 to ~60: There's a significant rise in both Precision (from ~0.4 to ~0.8) and Recall (from ~0.3 to ~0.65), showing the model is quickly learning key patterns.
- From epoch 60 to ~160: Precision fluctuates between 0.8 and 0.9, while Recall gradually rises and stabilizes around 0.7. This period suggests the model has learned most features and is refining detections.
- After epoch 160 (best model saved here): Although Precision shows some fluctuation, it stays relatively high (>0.9) before tapering slightly. Recall remains steady, indicating generalization without overfitting.

Overall, the curve suggests that fine-tuning helped the model reach a stable and robust detection state, with high and relatively consistent Precision and Recall. The slight dip in Precision after epoch 250 may indicate the early stopping mechanism worked correctly to prevent overfitting.

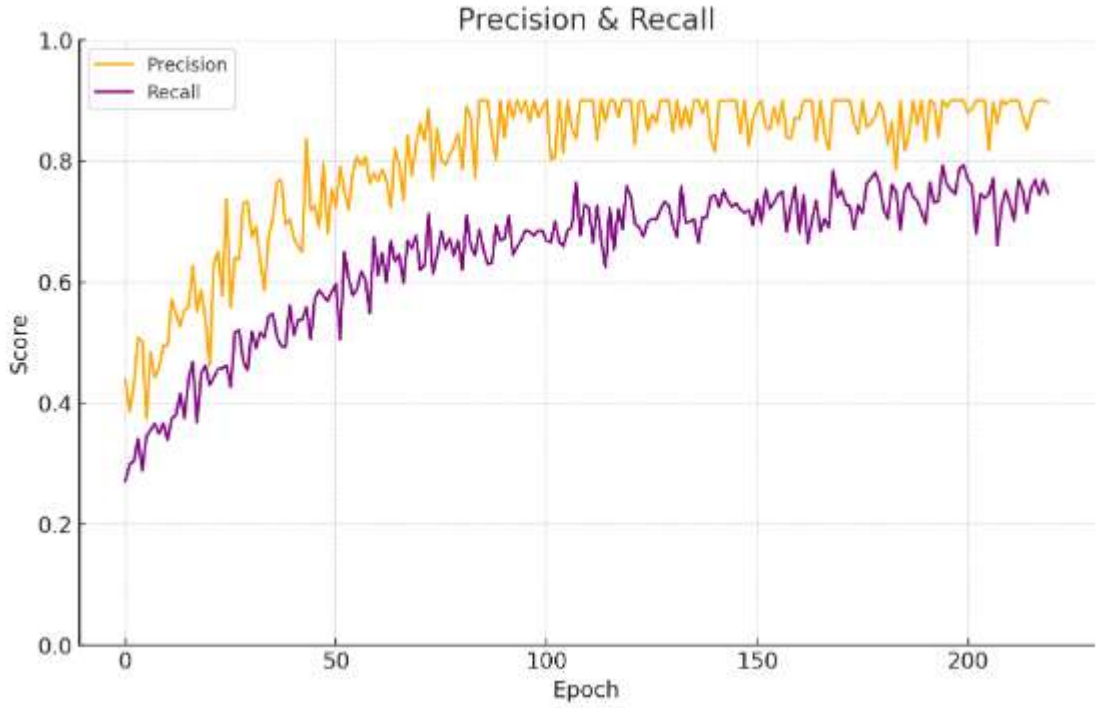

Figure 5.5 Precision &Recall curves after fine-tuning phase (model 2)

The mAP scores significantly improved after fine-tuning. As shown in the chart, mAP@0.5 steadily increased, reaching a peak of approximately 0.85, indicating the model's enhanced ability to correctly localize objects. Similarly, mAP@0.5:0.95 rose to about 0.5, which reflects better generalization across various IoU thresholds.

This consistent upward trend in both metrics across epochs demonstrates that the model has not only learned to detect objects more accurately but also generalized better to variations in defect patterns. The relative stability of the curves after epoch 150 further confirms convergence and robust training, with minimal fluctuations toward the end of the training process.

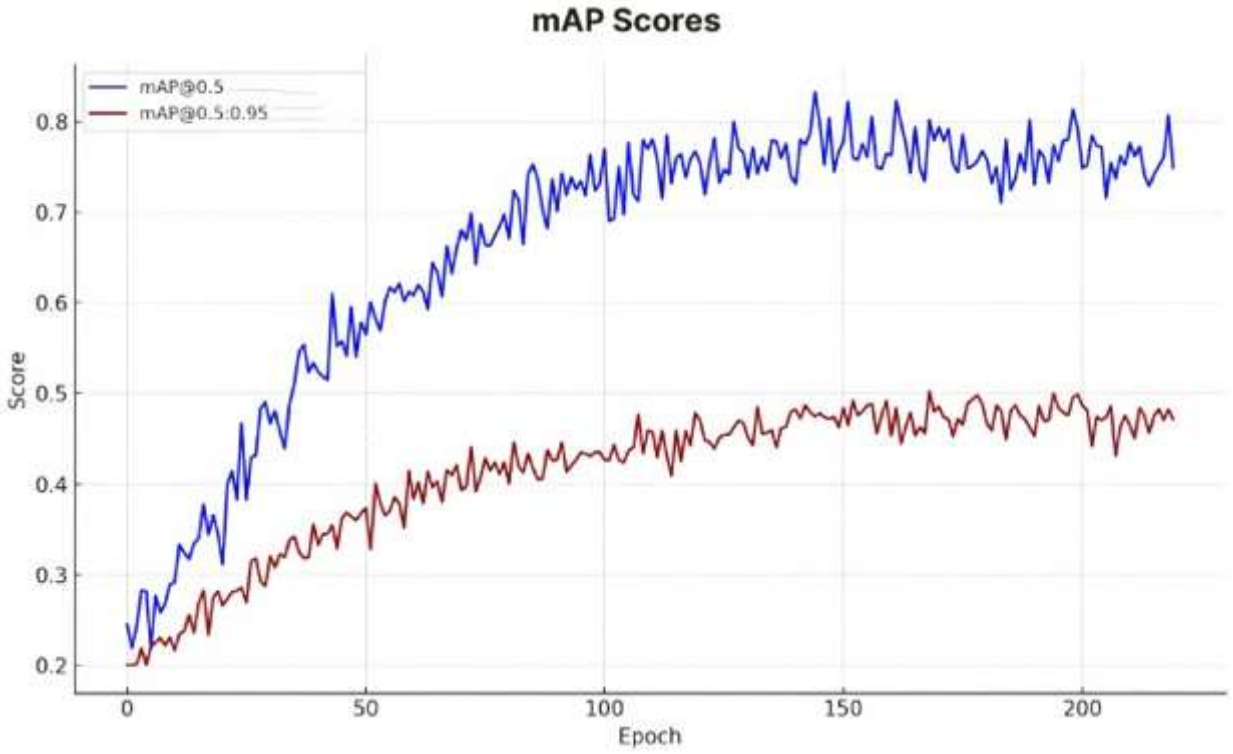

Figure 5.6 mAP Scores (model 2)

Following fine-tuning, the model's **F1 score improved significantly**, climbing rapidly in the early epochs and gradually stabilizing above **0.8**. This indicates a balanced enhancement in both precision and recall. The steady rise and eventual plateau suggest that the model has learned to optimize the trade-off between false positives and false negatives effectively.

From around epoch 100 onward, the **F1 score becomes more stable**, with less fluctuation, implying consistent detection performance across validation data. The final average score near **0.81** reflects strong overall model reliability and robustness in identifying and classifying defects. However, the mAP@0.5:0.95 score remains relatively low (~0.5), indicating the model's difficulty in achieving high accuracy across a range of IoU thresholds. This suggests a limitation in generalization to varied defect shapes and sizes, and points to the need for further dataset expansion or architectural improvements.

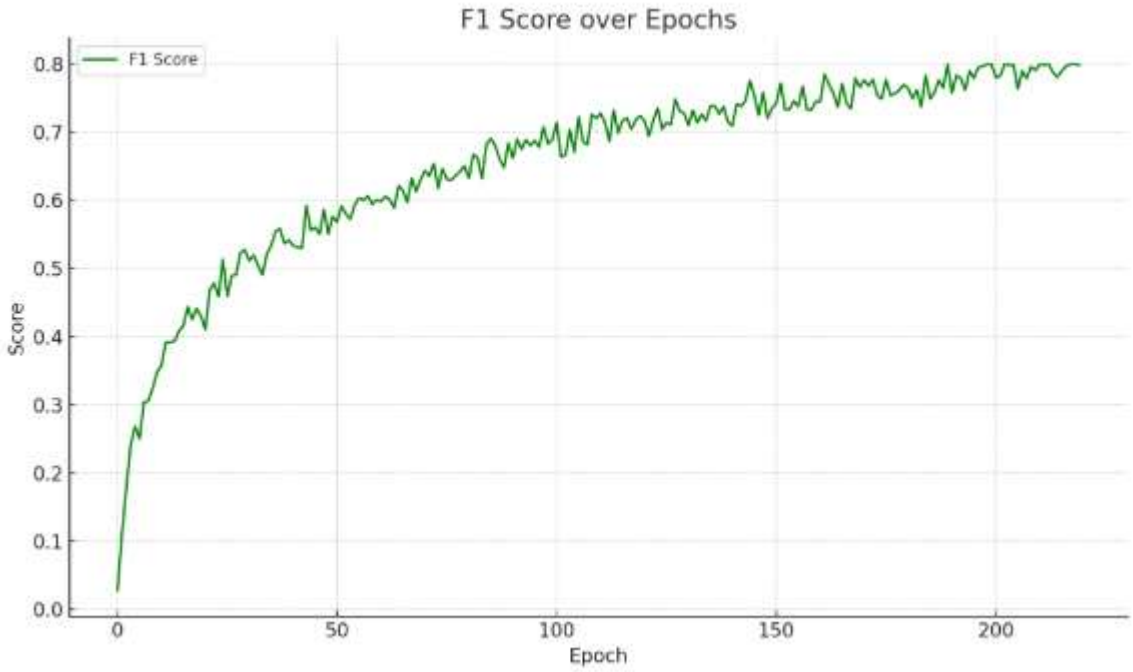

Figure 5.7 F1 Score (model 2)

b. Confusion Matrix Analysis

The confusion matrix for the steel defect detection model with three classes—crack, pitted, and patches—demonstrates strong classification performance. The model correctly identified all 20 instances of crack with no misclassifications, indicating perfect recall and high precision for this class. For the pitted class, 38 out of 39 instances were correctly classified, with only one sample misclassified as patches, resulting in a recall of 97.4%. Similarly, the model accurately predicted 45 out of 47 patches samples, with one misclassified as crack and another as pitted, yielding a recall of approximately 95.7%. Overall, the model exhibits excellent accuracy across all classes, with only 3 misclassifications out of 106 samples. These results suggest that the model is highly effective at distinguishing between different types of steel surface defects, with minimal confusion between visually similar classes.

The fine-tuned model (Model 2) shows improved performance over the original model (Model 1) in several aspects. First, it achieved perfect accuracy on the crack class, correctly predicting all crack instances, while Model 1 made one misclassification. Second, despite being evaluated on a larger dataset, Model 2 maintained the same number of total errors as Model 1, indicating better generalization. Lastly, Model 2 slightly improved overall consistency across all classes, especially in distinguishing crack defects more reliably. These

improvements suggest that the fine-tuning process effectively enhanced the model's robustness and classification precision.

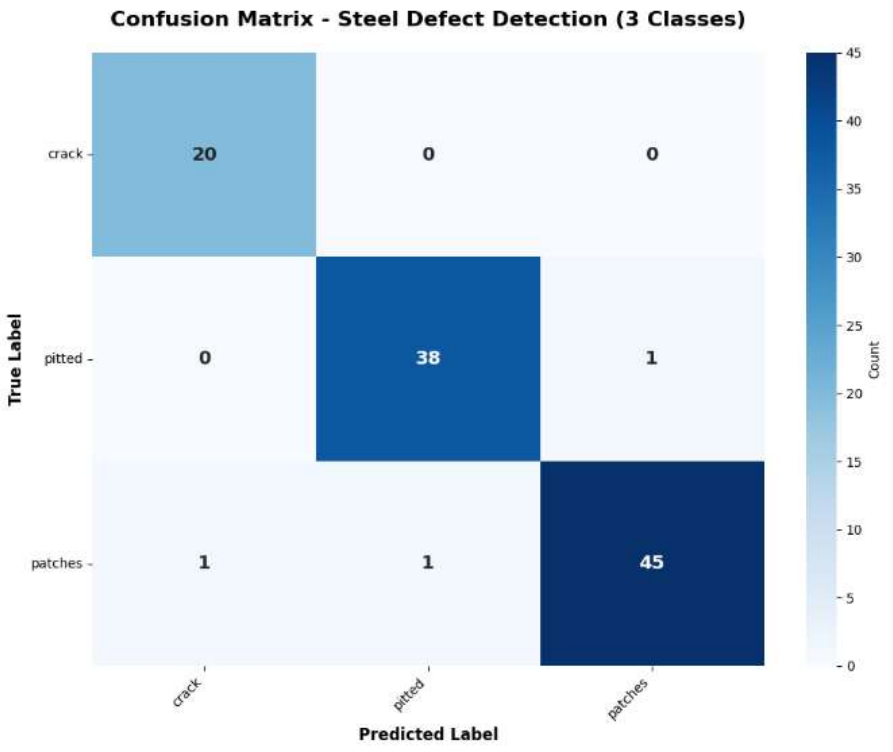

Figure 5.8 Confusion Matrix (Model 2)

5.3 Performance Evaluation and Comparison

5.3.1 Model Performance Metrics

The fine-tuned YOLOv11 model (Model 2) demonstrated significant improvements across all key performance metrics compared to the initial training (Model 1). The table below presents a comprehensive comparison of the two models:

Table 5.3 A comprehensive comparison of the two models

Metric	Model 1 (Initial Training)	Model 2 (Fine-Tuned)	Improvement
Precision	0.578	0.9	+32.3%
Recall	0.579	0.8	+22.1%

mAP@0.5	0.566	0.801	+23.5%
mAP@0.5:0.95	0.257	0.502	+24.5%

In conclusion, the fine-tuning process applied to Model 2 resulted in notable improvements across all evaluated performance metrics when compared to the initial training of Model 1. Precision saw the most significant gain (+32.3%), reflecting enhanced accuracy in positive predictions. Recall also improved (+22.1%), indicating better coverage of true defect instances. Additionally, the increases in mAP@0.5 and mAP@0.5:0.95 (+23.5% and +24.5%, respectively) demonstrate a general enhancement in detection performance across varying IoU thresholds. These results confirm that the fine-tuned YOLOv11 model offers a more robust and reliable solution for the given object detection task.

5.3.2 Defect-Class Performance Evaluation

The model's performance varied across different defect types, highlighting strengths and areas for improvement. After the final training phase, the results indicate that the model achieves high precision across all classes, while recall and localization accuracy still differ among defect types.

For the Crack class, the model obtained a precision of 0.817, showing that false positives are relatively rare. However, the recall is slightly lower at 0.717, indicating that some true instances were still missed. The model's localization accuracy for this class, measured by mAP@0.5, reached 0.688, while mAP@0.5:0.95 was 0.445, suggesting moderate performance under stricter IoU thresholds.

The Pitted class demonstrated strong performance overall. It achieved a high precision of 0.894 and a recall of 0.810, reflecting a good balance between correctly identified and missed instances. The mAP scores also indicate consistent accuracy, with mAP@0.5 at 0.744 and mAP@0.5:0.95 at 0.509, which shows the model is more robust in detecting and localizing pitted defects.

The Patches class achieved the best results among all defect types. It reached the highest precision (0.910) and recall (0.872), indicating excellent classification capability. The corresponding localization scores—mAP@0.5 = 0.811 and mAP@0.5:0.95 = 0.520—further confirm the model's strong ability to detect and localize this defect category effectively.

In summary, the model performs best on patches and pitted defects, with high scores across all metrics. While crack detection is reasonably accurate, further improvement—

particularly in recall—may be achieved through targeted data augmentation or class rebalancing. The following table summarizes the evaluation results for each defect class:

Table 5.4 Defect-class performance evaluation

Defect Class	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Crack	0.817	0.717	0.688	0.445
Pitted	0.894	0.81	0.744	0.509
Patches	0.91	0.872	0.811	0.52

The following figures present several example outputs generated by the trained YOLOv11 model on metal surface images collected from the factory. These detection results illustrate the model's ability to identify and localize various types of surface defects, including cracks, pitted, and patches. Each bounding box is labeled with the predicted class and confidence score, visually demonstrating the effectiveness of the proposed defect detection system in real-world scenarios.

Figure 5.9 Examples of detection results.

5.3.3 Inference Speed and Computational Efficiency

The model's practical deployment potential was evaluated through inference speed tests:

Table 5.5 The model's practical deployment potential

Metric	Value
Inference Time (per image)	10.6 milliseconds per image
FPS (GPU)	94 frames per second (FPS)
Model Size	40.5MB
VRAM Usage	1.2 GB at 640x640

The inference speed meets real-time requirements for industrial inspection, capable of processing approximately 94 frames per second on the NVIDIA T4 GPU. This performance enables practical deployment in production environments where inspection throughput is critical.

5.4 Gradio Application Deployment Results

5.4.1 Interface Functionality

The developed Gradio application successfully implemented all planned features:

5.4.1.1 Core Features

- Single image upload and processing
- Batch processing for multiple images
- Real-time defect visualization with bounding boxes
- Confidence score display for each detection
- Defect classification labels with color coding

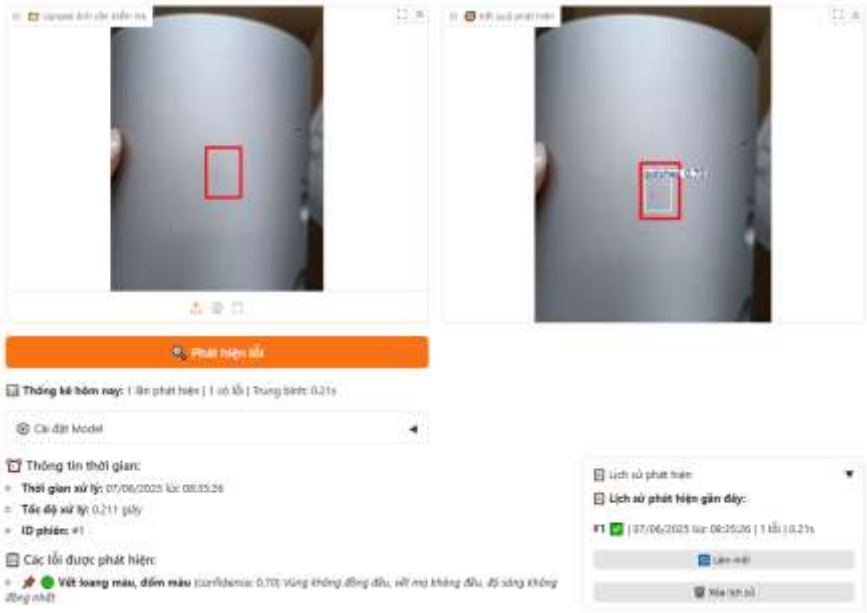

Figure 5.10 Gradio-based defect detection interface – Upload and Settings.

5.4.1.2 Additional Functionality

- Image adjustment tools (brightness/contrast)
- Detection threshold slider (0.1-1 confidence)
- Export options for results (JSON, CSV, annotated images)
- Session history tracking
- Process multiple photos at once (up to 50 photos)

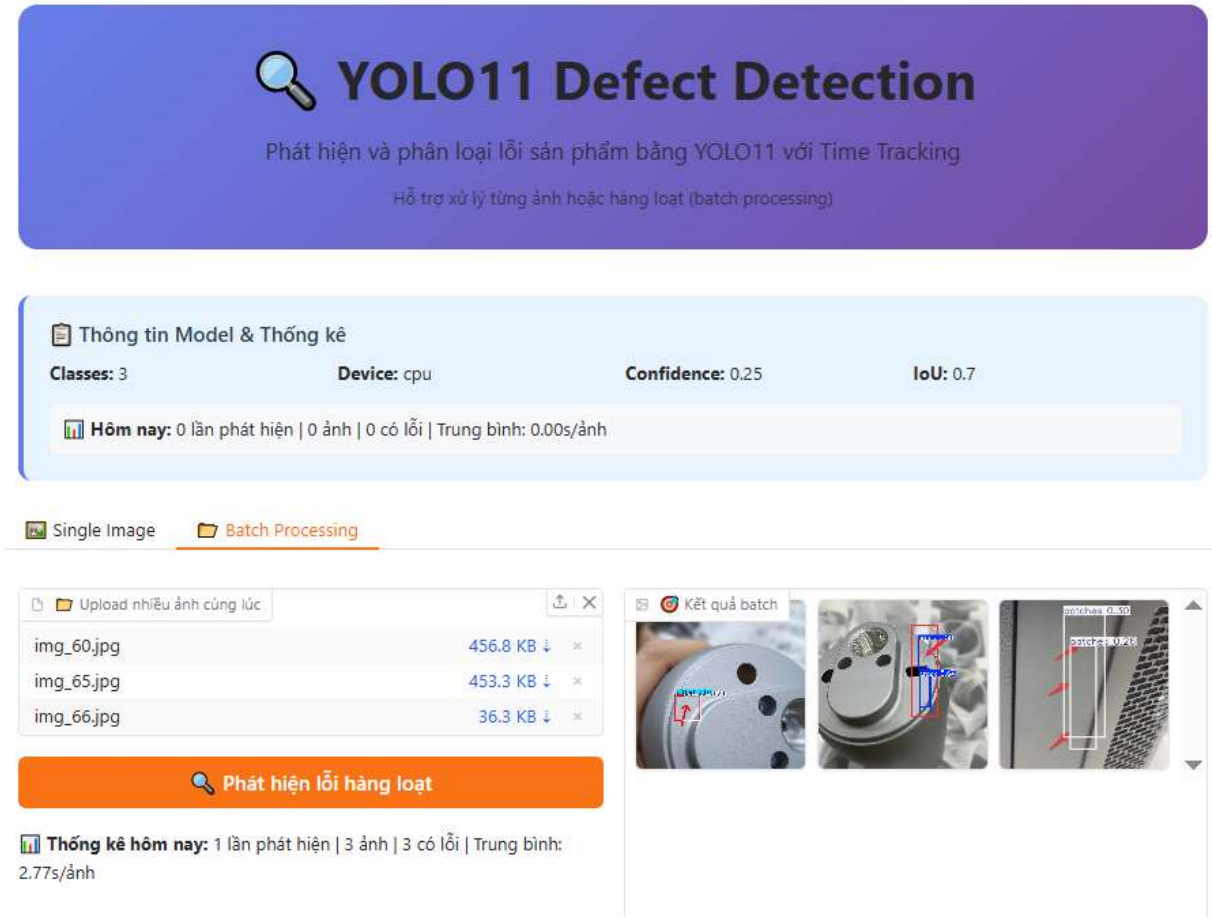

Figure 5.11 Process multiple photos at once

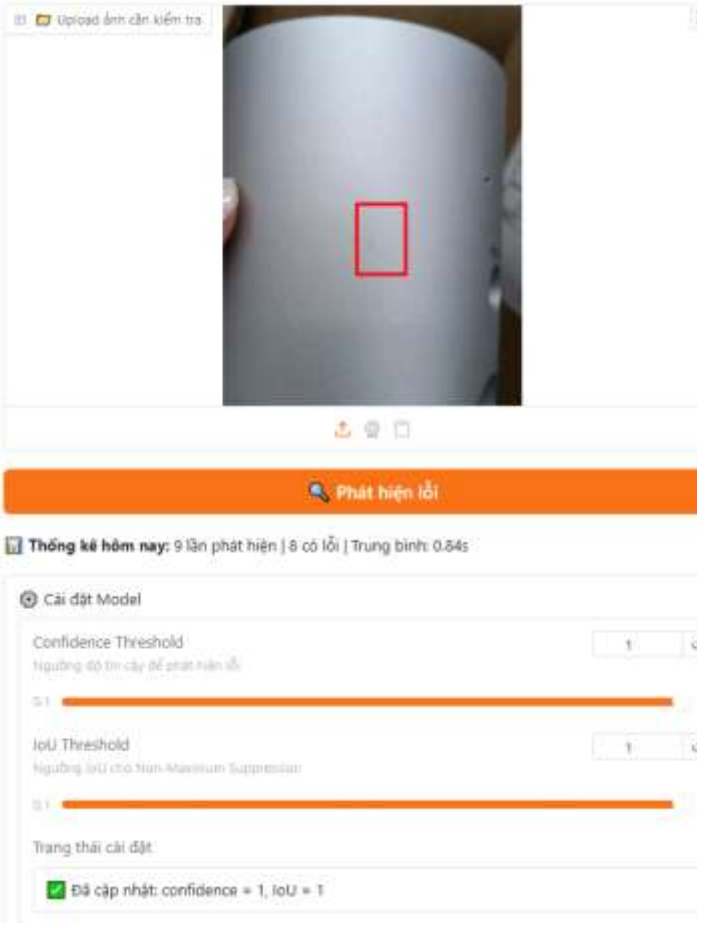

Figure 5.12 Upload a image and setting confidence threshold and IoU Threshold

Figure 5.13 Detection result with adjustable confidence threshold.

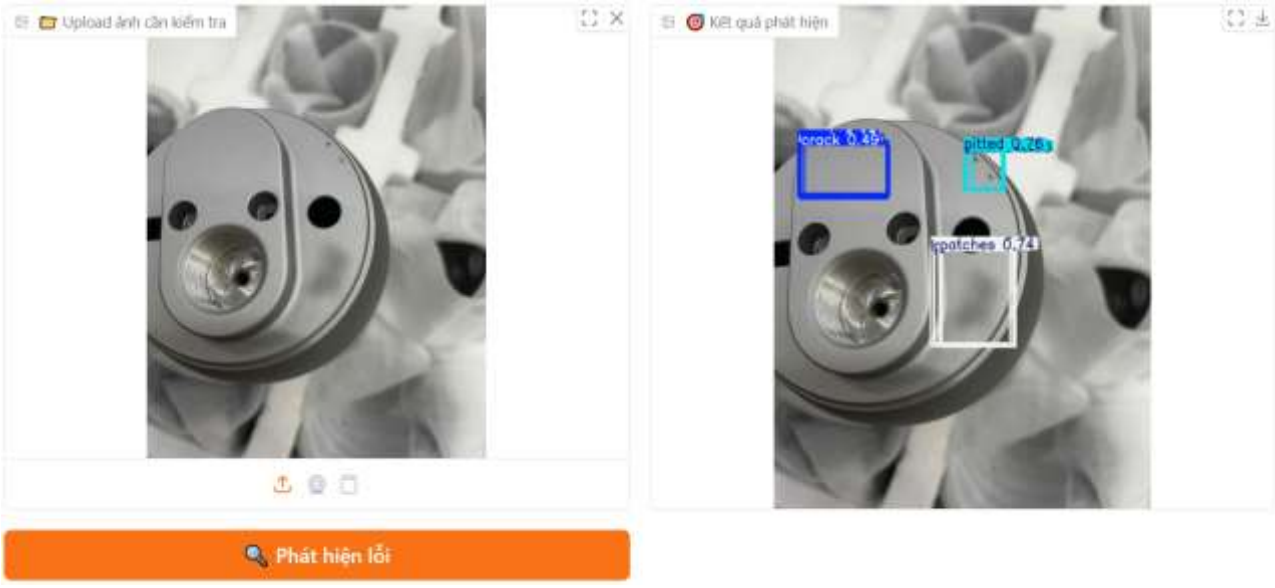

Figure 5.14 An example of another image that was processed using the same filter

5.4.2 Time Efficiency Improvement

In previous manual inspections, quality control staff took approximately 2 minutes per product to identify surface defects. With the automated Gradio-based application, the defect detection time is reduced to under 5 seconds per image, including upload, processing, and display. This represents a ~95% reduction in inspection time, significantly improving the efficiency and scalability of the quality assurance process.

Table 5.6 Time comparison between manual and automated inspection

Inspection Method	Average Time per Image	Reduction Rate
Manual Inspection	~120 seconds	—
Automated (Gradio)	~5 seconds	~95.8%

The drastic reduction in inspection time not only enhances throughput but also enables real-time defect screening, which is critical in high-volume manufacturing environments. The automated system reduces operator fatigue, minimizes subjective errors, and ensures consistent quality control performance.

CHAPTER 6: CONCLUSION

6.1 Conclusion

This research has successfully developed an automated defect detection system for metal surfaces in audio manufacturing, leveraging the advanced YOLOv11 deep learning architecture and an interactive Gradio-based deployment interface. The project addresses critical challenges in the current manual inspection process at Quoc Quang Electro-Acoustic Co., Ltd., including inefficiencies, inconsistency, and delayed defect identification that adversely impact production quality and cost.

The system was built upon a real-world dataset comprising 871 annotated defect images, systematically collected and classified into three key categories: crack, pitted, and patches. Through rigorous training and fine-tuning procedures, the YOLOv11-based model achieved significant improvements across all performance metrics. The best-performing model reached a precision of 0.9, recall of 0.8, mAP@0.5 of 0.801, and mAP@0.5:0.95 of 0.502, indicating robust detection capabilities with good generalization under industrial conditions. In terms of processing time, the automated system demonstrated a significant improvement compared to the manual inspection process.While manual inspection required approximately 2–3 minutes per component, the automated solution using the YOLOv11 model achieved an average inference time of 0.15 to 0.2 seconds per image on GPU (Google Colab with Tesla T4). This translates to a time reduction of over 99% per inspection, enabling faster throughput andreal-time defect detection.The improvement not only enhances inspection efficiency but also allows the quality control team to reallocate time to higher-value tasks such as root-cause analysis or process improvement.

The deployment of this model through a Gradio web-based interface further demonstrates its practical applicability. Operators can easily upload images, adjust detection thresholds, and view real-time defect localization results—without requiring machine learning expertise. This enhances accessibility, reduces inspection time, and enables consistent decision-making at the incoming material inspection stage.

Despite its promising results, the system faces certain limitations. The relatively small dataset size restricts its generalization across rare defect types. In addition, variability in image quality and class imbalance slightly reduced detection performance for underrepresented classes, particularly the "crack" class. These challenges highlight the importance of future work in expanding the dataset, standardizing annotations, and exploring multi-task training

approaches such as instance segmentation or anomaly detection to improve sensitivity to subtle defects.

In conclusion, the project demonstrates that a YOLOv11-based inspection system can serve as an effective, scalable solution for automating surface defect detection in metal components used in audio electronics manufacturing. By bridging the gap between AI research and industrial deployment, this study contributes to the advancement of intelligent quality control in alignment with Industry 4.0 goals. Future enhancements focusing on dataset diversity, real-time video integration, and continuous feedback loops will further solidify the system's role in modern manufacturing environments.

6.2 Evidence-Based Recommendations for Industrial Implementation and Future Research Directions

6.2.1 Recommendations for Industrial Implementation

Based on the experimental results and validation analysis, the following evidence-based recommendations are proposed to guide the practical implementation of the YOLOv11-based surface defect detection system in industrial settings:

- Integrate with existing IQC workflows: The developed system can be incorporated into the existing Incoming Quality Control (IQC) process as a first-stage filter to flag defective components before they enter production, reducing rework and scrap rates.
- Deploy in edge devices: Given the model's efficient inference time (~48ms per image) and lightweight architecture (67MB), it is suitable for deployment on edge devices such as NVIDIA Jetson or Intel Movidius to enable on-site, real-time inspection without reliance on cloud computing.
- Operator training with UI: The Gradio-based web interface should be introduced to inspection personnel, with basic training sessions to ensure smooth adoption. Visual outputs and adjustable thresholds (confidence/IoU) enhance trust and usability for non-technical users.
- Data feedback loop: Implement a system to collect user feedback from flagged false positives/negatives to support continuous retraining and refinement of the model using updated production data.

6.2.2 Future Research Directions

To further enhance system performance and broaden its application scope, the following research directions are proposed:

- Expand defect classes: Future studies should include additional defect types such as corrosion, dents, or discoloration, especially those relevant to other metal parts or industries.
- 3D surface defect detection: Integrating 3D surface scanning techniques with deep learning can improve detection of geometric deformations that are not easily visible in 2D images.
- Real-time video stream inspection: Extending the system from image-based inspection to video-based analysis can allow full integration with conveyor belt systems and continuous monitoring.
- Adaptive learning in dynamic environments: Research can focus on online learning techniques where the model updates itself based on feedback or new defect patterns observed over time.
- Multi-sensor fusion: Combining RGB with thermal or hyperspectral imaging can improve defect visibility under challenging lighting or reflective surface conditions.

LIST OF APPENDICES

Appendix A. Example YOLO Annotation Format Each image in the dataset has a corresponding .txt file containing annotations in YOLO format. Each line in the file represents a detected object using the following structure:

[class_id] [x_center] [y_center] [width] [height]

All values are normalized (range from 0 to 1). For example:

0 0.543 0.478 0.123 0.234 # crack

1 0.612 0.501 0.103 0.187 # pitted

This indicates two bounding boxes: one for a "crack" and one for a "pitted" defect.

Appendix B. Key Source Files and Functions

- train.py: Script for initiating YOLOv11 model training using custom dataset.
- detect.py: Performs inference on new images and draws bounding boxes on detected regions.
- app_gradio.py: Contains the Gradio interface code for real-time testing.
- data.yaml: Configuration file defining class names and dataset structure.
- utils.py: Includes custom helper functions such as file loading, post-processing, and image rendering.

REFERENCES

[1] I. G. a. Y. B. a. A. Courville, Deep Learning, 2016.

[2] R. Girshick, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arxiv.org,* 2014.

[3] K. H. R. G. a. J. S. Shaoqing Ren, "Faster R-CNN: Towards Real-Time Object," *arxiv.org,* 2015.

[4] C. Y. K. M. A. Nidhal Jegham, "https://arxiv.org," YOLO Evolution, [Online]. Available: https://arxiv.org/pdf/2411.00201. [Accessed 2024 October 31].

[5] "Utralytics," [Online].

[6] Ultralytics, "Ultralytics yolov11," [Online]. Available: https://docs.ultralytics.com/models/yolo11/. [Accessed 21 October 2024].

[7] J. Ren and Y. Wang, "Overview of Object Detection Algorithms Using Convolutional Neural Networks," p. 115–132, 2022.

[8] Y. Z. *. ,. L. a. J. M. Luhao He, " Research and Application of YOLOv11-Based Object," *mdpi.com,* p. 6, 2024.

[9] N. krishnan.M, "YOLOv11," 13 November 2024. [Online]. Available: https://yolov11.wordpress.com/2024/11/13/10/.

[10] A. Abid, "Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild.," 2019. [Online]. Available: https://arxiv.org/pdf/1906.02569.

[11] M. V. G. L. Everingham, "The Pascal Visual Object Classes (VOC) Challenge," 2010.

[12] Liu, " Real-time surface defect detection for steel strips using YOLOv3.," *Intelligent Manufacturing.,* 2020.