An Industrial Oriented Mini Project Report

on

Automated Steel Surface Defect Detection Using Deep Learning

submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

by

22WH1A05E3	**Ms. G AKSHARA**
22WH1A05E5	**Ms. V SWATHI**
22WH1A05H8	**Ms. G SRUJANA**

under the esteemed guidance of

Dr. M. Shanmuga Sundari
Assistant Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD College of Engineering for Women
(Approved by AICTE | Affiliated to JNTUH)
(NAAC Accredited – A Grade | NBA Accredited B. Tech. (EEE, ECE, CSE and IT))
Bachupally, Hyderabad -500090

June, 2025

BVRIT HYDERABAD College of Engineering for Women
(Approved by AICTE | Affiliated to JNTUH)
(NAAC Accredited – A Grade | NBA Accredited B. Tech. (EEE, ECE, CSE and IT))
Bachupally, Hyderabad -500090
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Industrial Oriented Mini Project entitled "**Automated Steel Surface Defect Detection Using Deep Learning**" is a bonafide work carried out by **Ms. G AKSHARA (22WH1A05E3), Ms. V SWATHI (22WH1A05E5), Ms. G SRUJANA (22WH1A05H8)** in partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering**, **BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, under my guidance and supervision. The results embodied in this Industrial Oriented Mini Project work have not been submitted to any other University/Institute for the award of any Degree/Diploma.

Internal Guide
Dr. M. Shanmuga Sundari
Assistant Professor, CSE

Head of the Department
Dr. M Sree Vani
Professor, CSE

External Examiner

DECLARATION

We hereby declare that the work presented in this Industrial Oriented Mini Project entitled **"Automated Steel Surface Defect Detection Using Deep Learning"** submitted towards completion of Industrial Oriented Mini Project work in III Year II Semester of B.Tech. in CSE at **BVRIT HYDERABAD College of Engineering for Women,** Hyderabad is an authentic record of our original work carried out under the guidance of **Dr. M. Shanmuga Sundari, Assistant Professor, Department of CSE.**

Ms. G . AKSHARA
(22WH1A05E3)

Ms. V. SWATHI
(22WH1A05E5)

Ms. G . SRUJANA
(22WH1A05H8)

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K.V.N. Sunitha**, **Principal, BVRIT HYDERABAD College of Engineering for Women,** for her support by providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr. M Sree Vani, HoD, Department of CSE, BVRIT HYDERABAD College of Engineering for Women,** for all timely support and valuable suggestions during the period of our project.

We are extremely thankful to our Internal Guide, **Dr. M. Shanmuga Sundari, Assistant Professor, CSE, BVRIT HYDERABAD College of Engineering for Women,** for her constant guidance and encouragement throughout the project.

Finally, we would like to thank our Industrial Oriented Mini Project Coordinator, all the faculty members and staff of the CSE department who helped us directly or indirectly. Last but not least, we wish to acknowledge our **Parents** and **Friends** for giving moral strength and constant encouragement.

Ms. G . AKSHARA
(22WH1A05E3)

Ms. V . SWATHI
(22WH1A05E5)

Ms. G . SRUJANA
(22WH1A05H8)

ABSTRACT

Steel surface defects such as cracks, dents, and scratches pose significant challenges to the quality, durability, and performance of steel products. Undetected anomalies can compromise structural integrity, leading to failures in downstream applications and increased costs due to material waste or recalls. Consequently, accurate and efficient defect detection is a critical requirement in modern steel manufacturing processes. However, traditional manual inspection techniques are often inefficient, inconsistent, and not scalable to high-speed industrial environments.

To address these limitations, this project proposes an automated steel surface defect detection system powered by deep learning. We utilized the YOLOv11 (You Only Look Once, version 11) object detection model, selected for its capability to deliver high precision and rapid inference, making it well-suited for real-time inspection scenarios. A custom dataset comprising diverse steel surface anomalies—including scratches, cracks, pits, and inclusions—was curated using Roboflow and used to train the model.

The entire system was implemented in Python using the Google Colab environment, with GPU acceleration to facilitate faster model training and evaluation. Extensive validation and testing were conducted on separate image sets to assess the model's ability to accurately localize and classify surface defects. The results consistently demonstrated the model's robustness and effectiveness in identifying multiple defect types with minimal false detections.

This project validates the application of deep learning for automated visual inspection in industrial settings. By eliminating the need for manual inspection and reducing human error, the proposed system offers a scalable, accurate, and time-efficient solution that can be seamlessly integrated into manufacturing pipelines. Ultimately, it supports enhanced quality assurance and operational efficiency in the steel production industry.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
AI	Artificial Intelligence
DL	Deep Learning
CNN	Convolutional Neural Network
YOLO	You Only Look Once
mAP	mean Average Precision
GPU	Graphics Processing Unit
FPS	Frames Per Second
IoU	Intersection over Union
TP	True Positive
FP	False Positive
FN	False Negative
TN	True Negative
QC	Quality Control
CSV	Comma Separated Values
PR Curve	Precision-Recall Curve

INDEX

CHAPTER 1

INTRODUCTION

In the rapidly evolving world of industrial automation, maintaining high-quality standards in steel manufacturing is not only critical to ensuring structural integrity and product performance but also a key driver of market competitiveness. Steel, being a foundational material in industries such as automotive, construction, and aerospace, is expected to meet stringent quality norms. Among the many quality determinants, surface defects like cracks, inclusions, pits, and scratches significantly impact both the aesthetic appeal and mechanical properties of steel. These defects can lead to product failure, costly recalls, and damage to a company's reputation if left unchecked.

Traditionally, the process of defect detection has been labor-intensive, relying on manual inspection or outdated computer vision systems that lack accuracy, speed, and scalability. Human-based inspection is prone to fatigue and inconsistencies, while classical image processing methods often fail in dynamic industrial environments where defect shapes, lighting conditions, and textures vary greatly. This has paved the way for more intelligent, automated inspection systems powered by Artificial Intelligence (AI) and Deep Learning (DL).

With the rise of Industry 4.0, smart manufacturing systems increasingly incorporate deep learning models for predictive maintenance, quality inspection, and real-time decision-making. In this context, object detection algorithms such as YOLO (You Only Look Once) have gained prominence for their ability to detect and localize objects swiftly and accurately. YOLOv11, the latest in the YOLO series, offers even higher detection precision, faster inference, and better real-time performance, making it an ideal candidate for industrial defect detection systems.

This project, implemented entirely using Google Colab and deployed as an open-source demo on Hugging Face Spaces, proposes the development of an automated steel surface defect detection system using the YOLOv11 architecture..

1.1 Problem Statement

Steel surface defects present a major challenge in quality assurance for manufacturing industries. These defects can compromise the durability, safety, and performance of end products, particularly in sectors where material reliability is paramount. Despite the critical need for precise inspection, the current methods suffer from several drawbacks:

- Manual inspection is slow, inconsistent, and labor-intensive.
- Traditional machine vision systems struggle with real-time processing and fail to detect small, irregular, or overlapping defects.
- Existing AI models may perform well in accuracy but are often too computationally heavy for edge deployment.

Hence, there is a pressing need for a defect detection system that is not only highly accurate and robust across multiple defect classes but also lightweight and optimized for deployment in production-line environments.

1.2 Objectives

The primary aim of this project is to build a robust defect detection system using deep learning. The key objectives include:

- To address the limitations of manual inspection methods for steel surface defects, which are often time-consuming, inconsistent, and imprecise at high production speeds.
- To develop an automated defect detection system using deep learning, specifically targeting surface anomalies such as cracks, dents, and scratches.
- To utilize the YOLOv11 object detection model for its high accuracy and fast inference speed in detecting steel surface defects.
- To train the model on a custom steel defect dataset obtained from Roboflow, covering multiple types of surface defects.

- To implement and evaluate the system using Python on Google Colab with GPU acceleration for efficient model training and testing.
- To validate the model's performance through visual and metric-based results, confirming its effectiveness in real-time defect identification.
- To demonstrate the feasibility and scalability of deep learning-based visual inspection for quality control in steel manufacturing industries.

1.3 Existing Work

Surface defect detection using deep learning has received significant attention in recent years. Early approaches relied on classical computer vision methods involving filters, thresholding, and edge detection. While these methods are fast, they lack generalization and perform poorly under variable lighting and noise.

Subsequent research turned to Convolutional Neural Networks (CNNs), which provided better feature extraction and defect classification capabilities. However, these models typically required large datasets, lacked speed, and were not optimized for real-time applications. Models such as YOLOv11 have been employed for defect detection with improved results, but they still face limitations in detecting small defects, handling unbalanced datasets, and performing on edge devices.

Few models have explored incorporating attention mechanisms or custom loss functions tailored to defect characteristics. Furthermore, optimization techniques such as pruning and quantization are rarely applied, which hinders deployment in constrained environments like smart factories or mobile devices. These gaps motivate the development of a more efficient and adaptive solution.

1.4 Proposed Work

This project presents a real-time, deep learning-based system for detecting steel surface defects. It aims to automate the defect detection process, replacing manual inspection methods that are often slow, inconsistent, and unsuitable for high-speed manufacturing environments.

The key highlights of the proposed work include:

- **Model Selection**: The YOLOv11 object detection model is used for its high accuracy and fast inference speed, making it ideal for real-time industrial applications.
- **Dataset:** A custom steel surface defect dataset, containing various types of defects such as cracks, dents, and scratches, is sourced and prepared using Roboflow.
- I**mplementation:** The system is developed using Python and PyTorch in Google Colab, leveraging GPU support for faster training and evaluation.
- **Model Evaluation:** The trained model is validated and tested on separate image sets, with visual outputs confirming its ability to detect and localize different defect types effectively.
- **Industrial Relevance:** This solution demonstrates the feasibility of using deep learning for automated quality inspection in steel manufacturing and can be scaled for integration into smart factories.

The model is trained on a labeled steel surface defect dataset prepared using Roboflow, which includes various types of anomalies such as cracks, dents, and scratches. Implementation is carried out using Python and PyTorch on Google Colab, leveraging GPU support for efficient training and evaluation.

To make the system accessible for public testing and demonstration, an open-source version is deployed on Hugging Face Spaces. The final system is designed to be scalable and adaptable, making it suitable for real-time deployment in industrial environments and aligning with the goals of Industry 4.0.

1.5 Importance of Quality Control in Industry

Quality Control (QC) plays a pivotal role in industrial manufacturing by ensuring that products meet specified standards and customer expectations. In sectors such as steel manufacturing, where safety, durability, and consistency are paramount, the presence of surface defects—such as cracks, inclusions, or scratches—can severely impact the functionality, appearance, and market value of the final product.

Modern consumers and industrial clients demand materials that are not only structurally sound but also visually flawless. Even minor deviations from quality standards can lead to product rejections, costly recalls, reputational damage, and legal liabilities. As a result, maintaining strict quality control throughout the production pipeline is essential for sustaining business competitiveness and operational efficiency.

Some of the key reasons why quality control is critically important in industries like steel manufacturing include:

- Safety Compliance: Defective steel products used in buildings, vehicles, or machinery can pose serious safety hazards. QC ensures that only safe, tested materials are released into the market.

- Minimizing Material Wastage: Early detection of defects allows faulty products to be rectified or removed before further processing, reducing resource wastage and saving manufacturing costs.

- Enhancing Customer Trust: Delivering consistent quality builds credibility with customers, encourages repeat business, and enhances brand reputation.

- Meeting Regulatory Standards: Many industries are governed by international and national quality standards (such as ISO, BIS, ASTM). Robust QC systems help companies comply with these mandatory requirements.

- Reducing Downtime and Rework: Defect detection at the right stage helps prevent downstream bottlenecks, ensuring a smoother and more predictable production process.

- Gaining Competitive Advantage: Organizations with advanced QC systems can guarantee better product performance, which gives them an edge over competitors in global markets.

1.6 Challenges in Manual Defect Detection

- **Inconsistency and Human Error:** Human inspectors can interpret the same defect differently depending on their experience, focus, or fatigue level. This leads to inconsistent results, where one batch may pass inspection while another with similar defects may be rejected.

- **Low Precision in Detecting Fine Defects:** Small or subtle defects such as micro-cracks, pits, or surface scratches are often missed during visual inspection. These anomalies can be hard to spot, especially when lighting conditions or surface textures make them less visible.

- **Slow Inspection Speed:** Manual inspection is time-consuming and cannot keep pace with high-speed production lines. As product volumes increase, human inspectors become a bottleneck, slowing down overall manufacturing efficiency.

- **Lack of Standardization:** The absence of a consistent, measurable framework for identifying and categorizing defects can lead to subjective assessments. What one inspector considers a critical defect, another might ignore, resulting in quality mismatches.

- **Limited Data Collection and Traceability:**Manual inspection processes often do not record detailed information about defect types, locations, or frequencies. Without digital logs, it is difficult to trace issues back to their source or analyze trends for process improvement.

- **Poor Scalability:** As production increases, scaling manual inspection requires hiring and training more inspectors. This increases operational costs and introduces further variability, making the system inefficient and unsustainable at larger scales.

- **Inspector Fatigue Over Time:** Repetitive inspection tasks lead to mental fatigue and reduced attention spans, especially during long shifts. This diminishes the inspector's ability to identify defects accurately, increasing the chance of errors or oversights.

- **Physical Safety Risks:** Inspectors may have to examine large, heavy, hot, or moving materials in potentially hazardous environments. These conditions pose safety risks and limit how thoroughly the inspection can be conducted.

- **Lack of Real-Time Feedback and Automation:** Manual detection does not provide instant identification of defects or integration with automated systems. This delays corrective actions and makes it impossible to halt production lines based on immediate quality issues.

CHAPTER 2

LITERATURE WORK

In this chapter, we explore the foundational studies and existing methodologies related to steel surface defect detection. The aim is to understand the progress made so far in this field and identify the strengths and limitations of current approaches. With the growing application of artificial intelligence and deep learning in industrial automation, numerous researchers have proposed innovative techniques to improve accuracy, speed, and scalability of defect detection systems. This chapter also highlights the technological tools employed and the specific challenges yet to be addressed, setting the stage for the proposed work in this project.

2.1 Related Work

Surface defect detection in steel manufacturing is a critical quality control task. Traditional visual inspection methods are labor-intensive and prone to human error. To address this, researchers have turned to deep learning techniques, particularly convolutional neural networks (CNNs) and object detection models like YOLO (You Only Look Once).

- Maojie Sun et al. (2024) presented a model compression approach for defect detection, leveraging expert knowledge and working conditions. By using pruning and quantization, their model became computationally efficient and suitable for real-time deployment without compromising accuracy.

- Tinglin Zhang et al. (2024) introduced GDM-YOLO, a YOLOv8s-based network that uses grouped depthwise convolution layers and improved anchor design. It outperformed many existing models on common datasets, particularly in detecting small defects and maintaining high precision.

- Fei Ren et al. (2024) proposed an enhanced YOLOv5-based model called ECA-SimSPPF-SIoU-YOLOv5. It integrates efficient channel attention (ECA), a simplified spatial pyramid pooling (SimSPPF), and SIoU loss for improved localization and classification. This model showed better performance in complex industrial scenarios.

These studies demonstrate that deep learning models, especially those adapted to industrial environments through lightweight and attention-based improvements, are highly effective for defect detection tasks.

2.2 Research Gaps

Despite significant advancements, several challenges remain in steel surface defect detection using deep learning:

- **Lack of Real-Time Deployment:** Many proposed models are computationally intensive and not optimized for real-time monitoring on edge devices in factories.
- **Difficulty Detecting Minor Defects:** Small or subtle surface anomalies are often missed by standard models, affecting overall quality control.
- **Generalization Across Conditions:** Models trained in one factory setting may not perform well under different lighting, surface textures, or environmental conditions.
- **High False Positive Rates:** Some models detect non-defect areas as defects, leading to unnecessary rework and increased operational costs.
- **Limited Feedback Adaptability:** Current models don’t incorporate feedback mechanisms for self-improvement based on real-time input data.

These gaps highlight the need for a more adaptive, lightweight, and scalable solution for real-world deployment.

2.3 Tools and Technologies

To build a robust and real-time steel surface defect detection system, the following tools and technologies are used:

- **YOLOv11:** State-of-the-art deep learning object detection models known for their speed and accuracy. These are the backbone of the proposed system.
- **Python:** Programming language used for implementing model training, evaluation, and deployment.
- **Ultralytics YOLO:** Deep learning frameworks used to build, train, and fine-tune models.
- **OpenCV:** For image processing and visualization of defects.
- **Google Colab:**Development environment used for coding, training, and experimentation.
- **Dataset:** Publicly available steel surface defect datasets such as NEU-DET or custom factory image datasets
- **Roboflow:** Used to prepare and manage the custom steel surface defect dataset, containing multiple classes of anomalies such as cracks, dents, and scratches.
- **Hugging Face Spaces:** The trained model is deployed here for public testing and demonstration of defect detection capabilities.

2.4 YOLO Architecture Evolution (v1 to v11)

YOLOv1 The Original

Before YOLO, object detection relied on two-stage CNN-based models like R-CNN and Fast R-CNN, which were accurate but slow and computationally heavy. YOLO, introduced in 2016 by Joseph Redmon and Ali Farhadi, transformed object detection by introducing a single-stage approach that performs both bounding box prediction and object classification in one pass.

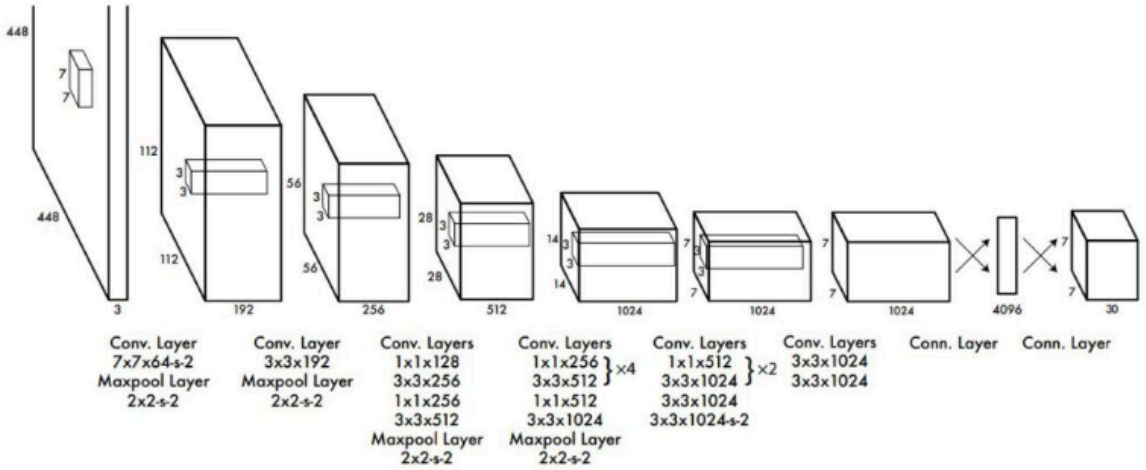


Fig-1.1- YOLOv1 architecture

The original YOLOv1 architecture featured 24 convolutional layers and 2 fully connected layers, inspired by GoogLeNet. This design allowed real-time detection by extracting features and making predictions in a single step. Key optimizations included:

- **Leaky ReLU activation** to prevent inactive neurons during training.
- **Dropout regularization** after the first fully connected layer to reduce overfitting.
- **Data augmentation** techniques to improve model generalization.

YOLOv2 (YOLO9000)

YOLOv2, also known as **YOLO9000**, was introduced in 2017 to overcome limitations in object detection datasets and improve accuracy and generalization. It could detect over **9000 object categories** by combining detection and classification tasks using a novel **joint training approach**.

Key improvements in YOLO9000 included:

- **Hierarchical Classification:**
  Implemented using the **WordTree structure**, allowing the model to generalize better and recognize a wider range of objects—even unseen ones.

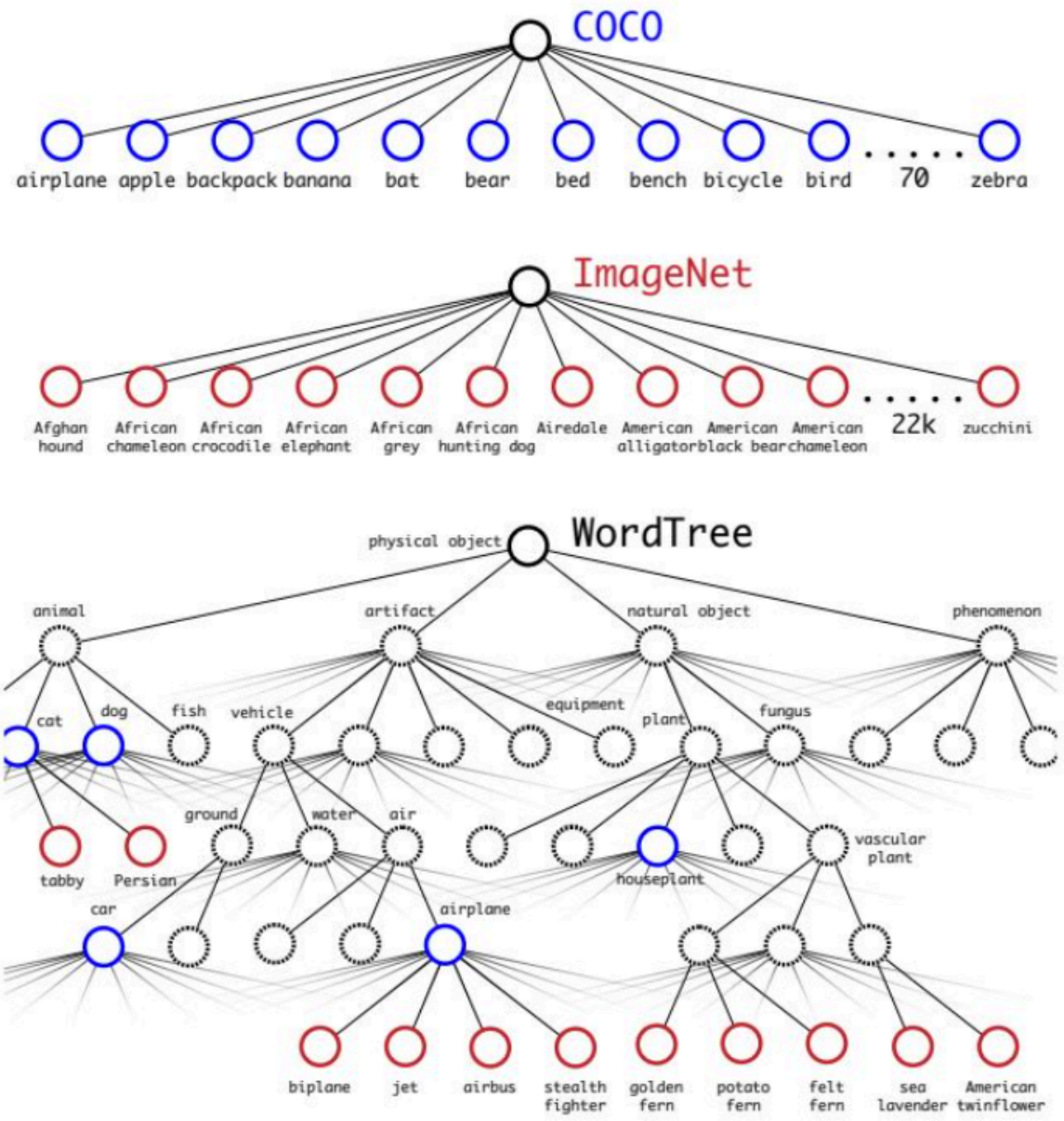


Fig-1.2- YOLO9000 combines datasets using the WordTree hierarchy

- **Architectural Enhancements:**

  Introduced **batch normalization** for better training stability and **anchor boxes** for improved localization. It used **Darknet-19** as its backbone—a fast and efficient 19-layer CNN.

- **Joint Training Algorithm:**

  Trained simultaneously on object detection (e.g., COCO) and image classification (e.g., ImageNet) datasets, enhancing both localization precision and vocabulary size.

YOLOv3

YOLOv3, released a few years after YOLO9000, introduced a series of **small but impactful changes** that significantly improved both **accuracy and speed**, making it one of the most balanced object detection models of its time.

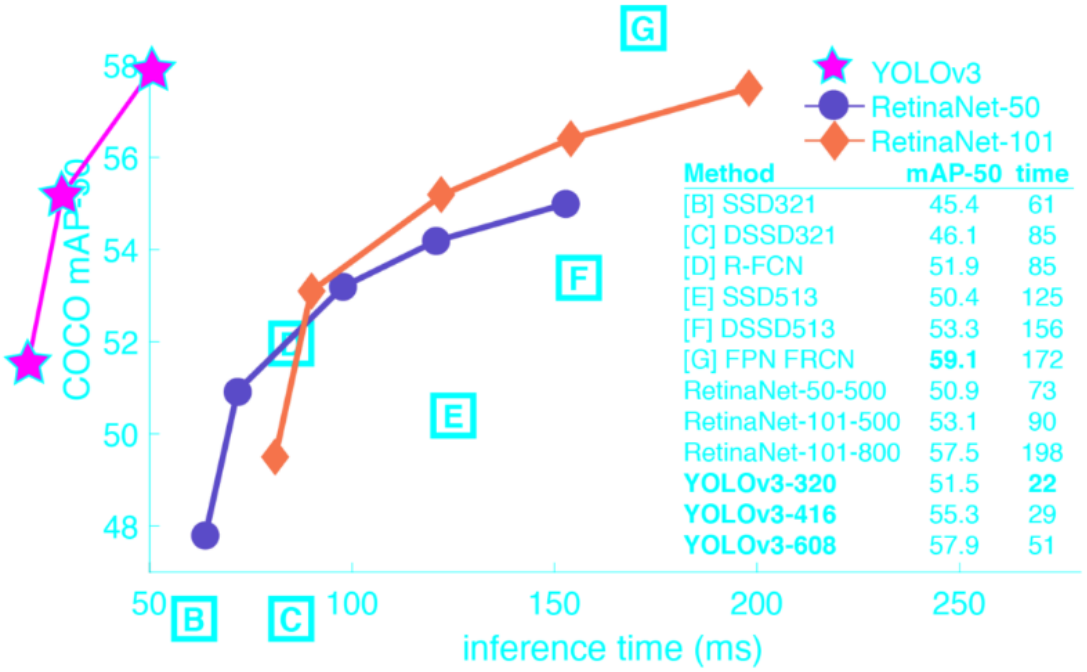


Fig-1.3 YOLOv3 compared to other state-of-the-art models at the time.

Key improvements in YOLOv3 include:

- **Multi-Scale Predictions:**
  YOLOv3 predicts bounding boxes at **three different scales**, inspired by feature pyramid networks. This helps the model detect **small, medium, and large objects** more effectively.
- **Improved Backbone – Darknet-53:**
  YOLOv3 uses **Darknet-53** as its backbone, a 53-layer CNN combining ideas from Darknet-19 and ResNet architectures. It offers better accuracy and efficiency compared to ResNet-101 and ResNet-152.
- **Logistic Classifiers:**
  Unlike previous versions that used softmax, YOLOv3 uses **independent logistic classifiers**, allowing each bounding box to predict **multiple labels**.

- **Training on COCO Dataset:**
  The model was trained only on the **COCO dataset**, enhancing general performance on a wide range of object categories.
- **Minor Fixes:**
  A small bug in YOLOv2's data loading was fixed, resulting in a modest **2-point mAP** gain.

YOLOv4 Optimization is Key

YOLOv4 marked a significant step in the YOLO family by refining the model's architecture and introducing extensive optimization techniques, making it a powerful balance of **speed and accuracy** for real-time object detection.

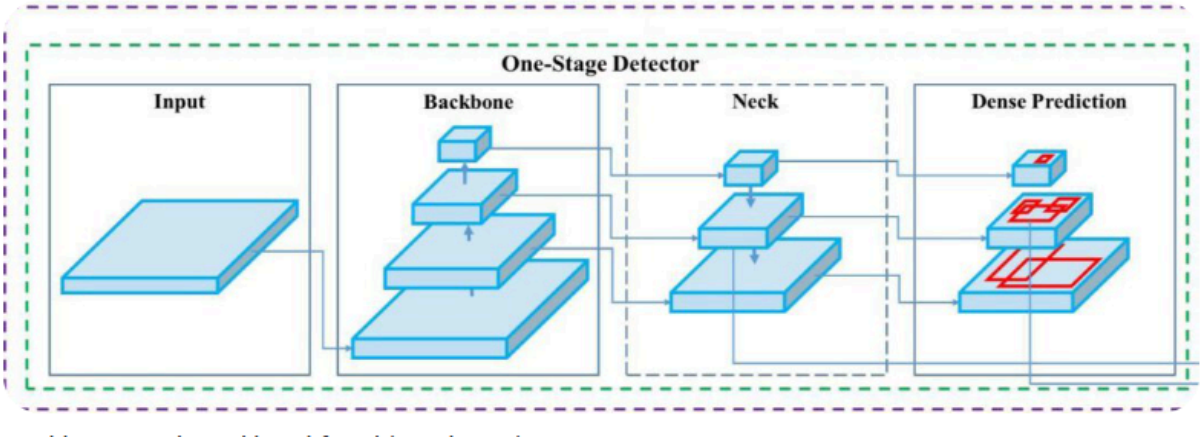


Fig-1.4 YOLOv4 architecture

- **Backbone:** Used **CSPDarknet53**, a version of Darknet-53 with Cross Stage Partial Network (CSPNet) to improve gradient flow and reduce computation without sacrificing accuracy.
- **Neck:** Incorporated **Spatial Pyramid Pooling (SPP)** and **Path Aggregation Network (PAN)** to aggregate features from multiple scales, enhancing object localization and robustness.
- **Head:** Maintained the **anchor-based head** architecture from YOLOv3, responsible for predicting bounding boxes, objectness scores, and class probabilities.

YOLOv5

Although YOLOv5 was not introduced through a formal research paper, it quickly became one of the most popular and developer-friendly models in the YOLO family. Released by Ultralytics just months after YOLOv4, YOLOv5 focused on ease of use, flexibility, and deployment readiness, making it highly accessible for real-world applications.

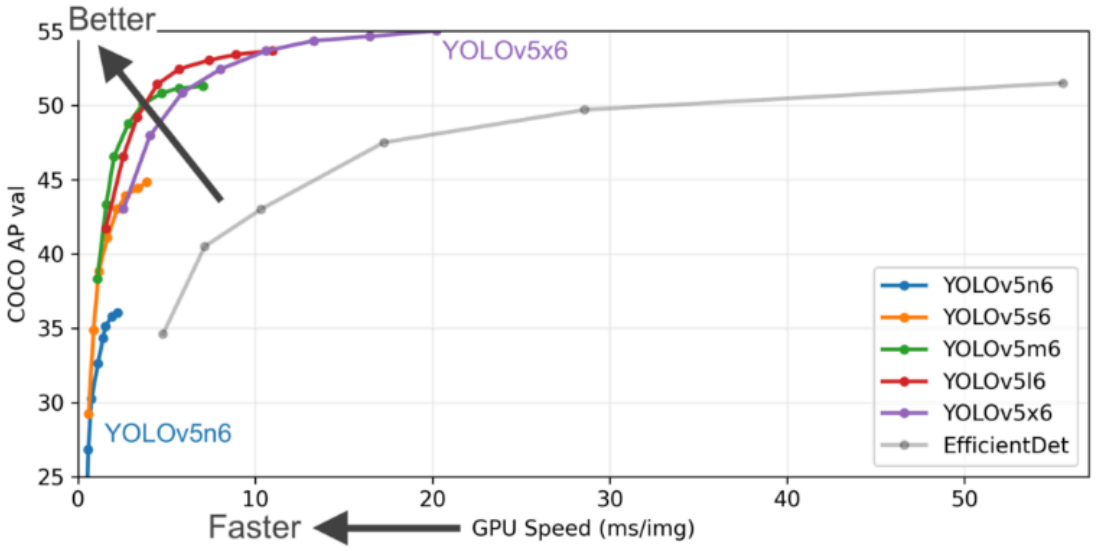


Fig-1.5 Performance of different YOLOv5 variations

- **PyTorch Implementation:**
  Unlike its predecessors, YOLOv5 is implemented in **PyTorch**, which simplifies model training, modification, and deployment for developers and researchers.
- **Developer-Friendly Design:**
  Comes with extensive **documentation**, multi-language support, and a clean API. This made YOLOv5 easier to train on custom datasets, even for beginners.
- **Export Flexibility:**
  Supports model export in multiple formats such as **ONNX**, **TorchScript**, and **CoreML**, allowing integration across platforms including mobile and edge devices.
- **Built-in Training Utilities:**
  Features useful training tricks like:
  i) **Test-Time Augmentation (TTA)** – Improves prediction accuracy by averaging multiple augmented versions of the input.
  ii) **Model Ensembling** – Combines predictions from multiple trained models to enhance robustness.

- **Lightweight Variants:**
  Offers models in various sizes (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x), allowing users to balance **speed and accuracy** based on hardware constraint

YOLOv6

YOLOv6 brought a major shift in the YOLO series by placing a strong emphasis on industrial use cases. It introduced significant architectural and training-level enhancements to balance real-time speed, deployment efficiency, and detection accuracy—especially on widely available hardware like the Tesla T4 GPU.

- **Backbone – EfficientRep**:
  Designed using hardware-aware CNN blocks, RepBlock for small models (N and S) CSPStackRepBlock for larger models (M and L). These are optimized for low-latency and high-efficiency feature extraction.
- **Neck – Rep-PAN:**
  An enhancement of PAN (used in YOLOv4/v5), Rep-PAN improves multi-scale feature fusion by using RepBlocks and CSPStackRepBlocks, increasing the network's ability to aggregate semantic information.

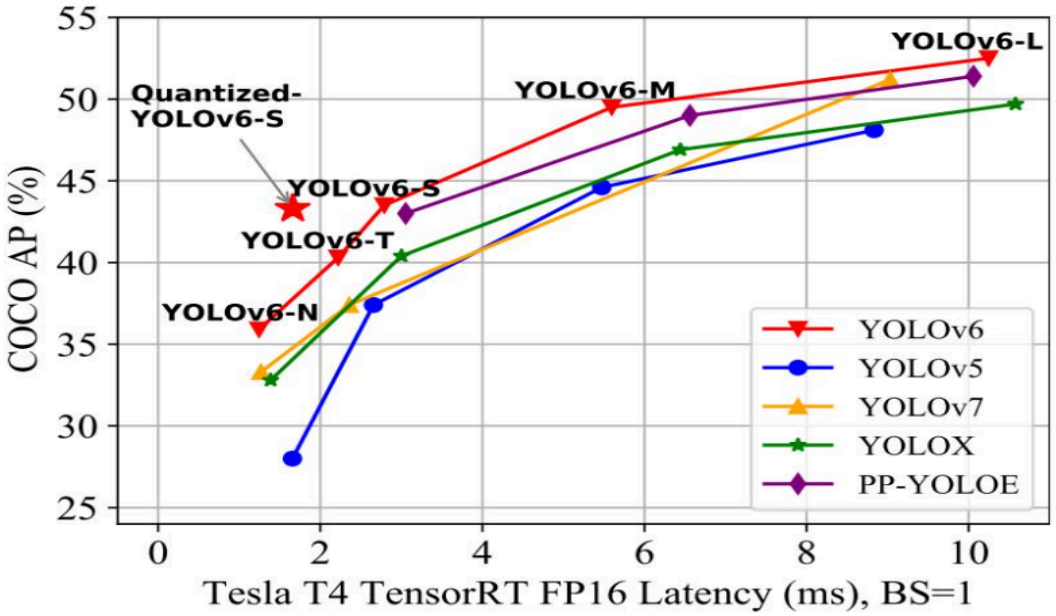


Fig-1.6 Comparison of Yolov6 with other efficient object detectors.

YOLOv7

YOLOv7 introduced a novel concept calling it the trainable bag of freebies (BoF). This includes a series of fine-grained refinements rather than a complete overhaul.

These refinements are primarily focused on optimizing the training process and enhancing the model's ability to learn effective representations without significantly increasing the computational costs. Following are some of the key features YOLOv7 introduced.

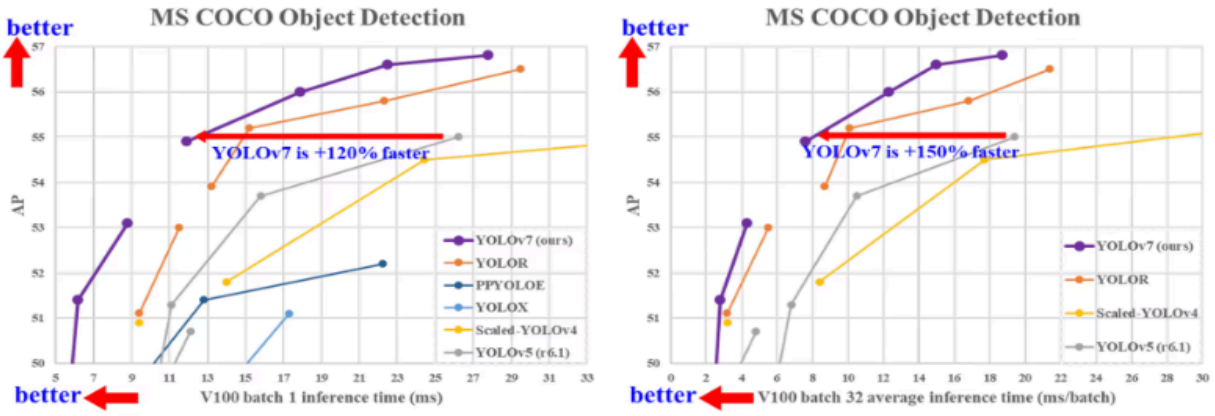


Fig-1.7 Comparison of Yolov7 with other efficient object detectors.

- **Trainable Bag of Freebies (BoF):** Introduced a set of training-time techniques to improve performance **without increasing inference cost**.
- **Model Re-parameterization:** Used during training to enhance gradient flow and optimize layers for **better representation learning**.
- **Dynamic Label Assignment:** Implemented **coarse-to-fine lead guided label assignment** to handle multi-output branches effectively.

YOLOv8

YOLOv8, developed by Ultralytics, is a powerful and user-friendly iteration in the YOLO series, designed for tasks like object detection, segmentation, classification, and pose estimation. It introduces an anchor-free split head architecture that enhances detection accuracy and efficiency. Despite lacking a formal research paper, YOLOv8 is widely adopted due to its ease of use, integration with the Ultralytics Python library, and strong performance across various benchmarks. It delivers state-of-the-art results with fast inference speeds, making it ideal for real-time applications.

Fig-1.8 Inference Image Of YoloV8

YOLOv9

YOLOv9 introduces a fundamentally new approach to object detection by directly addressing information loss and gradient reliability in deep neural networks. Unlike its predecessors,

YOLOv9 incorporates Programmable Gradient Information (PGI) and a new architecture called GELAN (Generalized Efficient Layer Aggregation Network) to enhance information flow and ensure effective learning.

PGI uses an auxiliary reversible branch to preserve full input data during forward passes, allowing for more informative and reliable gradients during backpropagation. Meanwhile, GELAN builds on the ELAN design by optimizing gradient paths across computational blocks, improving both efficiency and prediction accuracy. Together, these innovations position YOLOv9 as a more explainable and performance-focused object detector in the YOLO family.

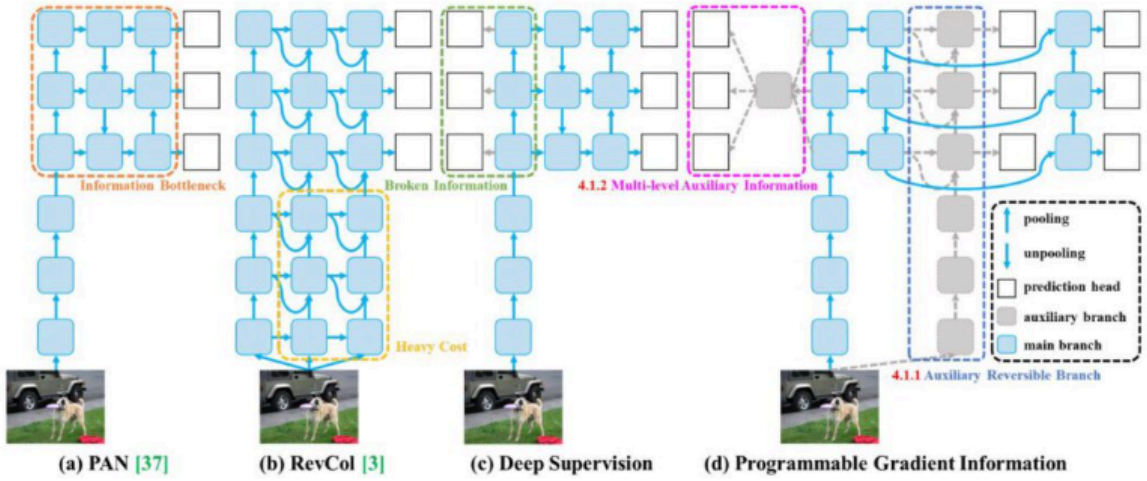


Fig-1.9 PGI and related network architectures and methods.

YOLOv10

YOLOv10 marks a significant breakthrough in real-time object detection by introducing **NMS-Free Detection**, which eliminates the traditional non-maximum suppression step during inference. This advancement enhances both **inference speed** and **deployment simplicity**.

The model uses a dual label assignment strategy—combining one-to-one and one-to-many supervision—to ensure accurate predictions without relying on NMS. Additionally, YOLOv10 employs a **holistic efficiency-accuracy design**, incorporating innovations like a lightweight classification head, spatial-channel decoupled downsampling, and rank-guided

blocks. These enhancements allow YOLOv10 to achieve state-of-the-art performance across speed and accuracy benchmarks, making it ideal for real-world real-time applications.

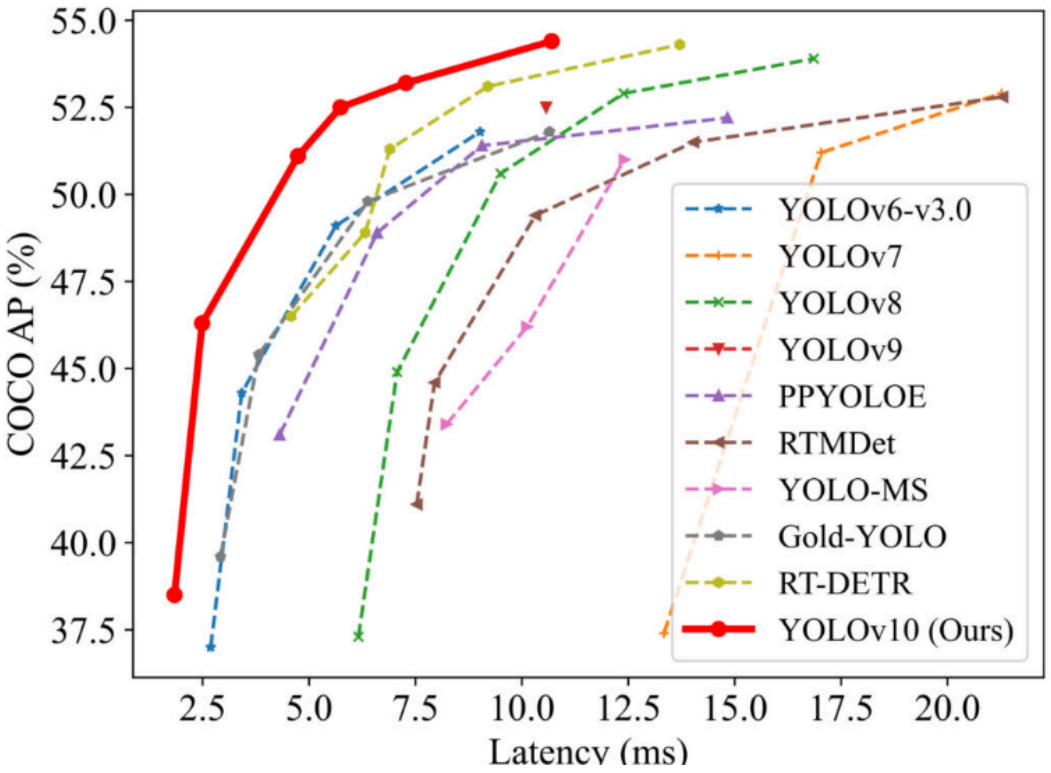


Fig-2.0- Comparisons with others in terms of latency-accuracy.

YOLOv11

YOLOv11, released in September 2024, introduced key architectural enhancements focused on computational efficiency and accuracy. It incorporates two major components—C3k2 and C2PSA blocks—that optimize feature extraction while reducing model complexity.

The C3k2 block is a lightweight version of the CSP bottleneck that replaces traditional convolutions with two smaller ones, significantly cutting down processing time. Meanwhile, the C2PSA block combines cross-stage partial connections with spatial attention, helping the model focus on critical regions in an image for more precise detection.

With these refinements, YOLOv11 achieves strong performance with fewer parameters, continuing the YOLO family's evolution in balancing speed and accuracy.

Model	Backbone	Anchor Free	mAP (%)	Inference Time (ms)	Model Size (MB)	Year
YOLOv3	Darknet-53	No	33.0	51	236	2018
YOLOv4	CSPDarknet53	No	43.5	29	245	2020
YOLOv5s	Custom (PyTorch)	No	36.8	17	27	2020
YOLOv7	E-ELAN	No	51.2	13	75	2022
YOLOv8n	Custom	**Yes**	37.3	**0.99**	6.2	2023
YOLOv10	GELAN + PGI	Yes	53.5	1.5	18	2024
YOLOv11	C3k2 + C2PSA	Yes	54.0	1.1	16	2024

Table-1 YOLO Model Comparison (Accuracy, Speed, Size)

CHAPTER 3

METHODOLOGY

This chapter provides a detailed overview of the methodology followed for developing a real-time steel surface defect detection system using deep learning. The focus is on automating the defect inspection process, which traditionally relies on manual inspection that is often slow, inconsistent, and lacks precision at high production speeds.

The proposed approach utilizes the **YOLOv11** object detection model, chosen for its high accuracy and fast inference speed, making it suitable for real-time industrial applications. A custom dataset containing various types of steel surface defects—such as cracks, dents, and scratches—was prepared using **Roboflow**.

The model was implemented using **Python** on **Google Colab**, leveraging GPU support for efficient training and evaluation. Once trained, the model was tested on a separate validation set to assess its performance. Visual inspection of results confirmed its ability to accurately detect and localize different types of defects.

Finally, the trained model was deployed on **Hugging Face Spaces** to allow public access and demonstration of its capabilities. This methodology supports a scalable and practical solution for automated visual inspection in modern steel manufacturing environments.

3.1 Proposed Model/Architecture

The core of the proposed system is a deep learning-based object detection pipeline that leverages the YOLOv11 model and a customized version of YOLOv5. The architecture is designed to balance detection accuracy with inference speed, making it suitable for real-time deployment in industrial environments.

Key Components of the Architecture:

- **Input Image Resizing**

  During training, all input images are resized to **200×200 pixels** using the `imgsz` parameter in YOLOv11. This step ensures uniform input dimensions, reduces training time, and minimizes memory usage—without significantly compromising the model's ability to learn meaningful features.

- **Dataset Integration via Roboflow**

  The dataset, named *"Augmented NEU Dataset"*, is integrated using the Roboflow API. Roboflow handles preprocessing, annotation conversion, and dataset structuring, providing separate folders for training, validation, and testing. It also auto-generates a `data.yaml` file that defines paths and class labels required by YOLOv11.

- **Backbone Model – YOLOv11**

  The core of the system is built on the **YOLOv11 medium model (`yolo11m.pt`)**, which is pre-trained and fine-tuned on the steel defect dataset. YOLOv11 is a state-of-the-art object detector that uses an anchor-free mechanism, enabling faster training and more flexible bounding box prediction. It also features improved feature fusion layers and simplified architecture for real-time deployment.

- **Training Engine**

  Training is performed using Ultralytics' command-line interface in detect mode. The model is trained for **50 epochs** on resized images, optimizing its weights to identify and localize defects. The training process uses YOLO's built-in loss functions, including objectness loss, classification loss, and bounding box regression loss.

- **Evaluation and Validation Module**

  The architecture includes built-in validation during training, and a dedicated validation phase afterward. It automatically generates confusion matrices, training curves, and detection overlays to visually inspect the model's learning performance.

- **Inference Pipeline**

  Once trained, the model is used in prediction mode to evaluate new test images. The

inference engine processes input images, runs detection, and outputs annotated images with bounding boxes, predicted class labels, and confidence scores—all saved automatically to designated folders.

- **Visualization Layer**

  Multiple outputs such as `train_batch0.jpg`, `val_batch0_pred.jpg`, `confusion_matrix.png`, and `results.png` are produced to visualize performance. These images provide a qualitative understanding of how well the model is detecting and classifying defects.

- **Export and Deployment Readiness**

  The best-trained model (`best.pt`) is downloaded for deployment purposes. While not deployed in this phase, the exported model can be hosted on platforms like **Hugging Face Spaces**, or integrated into desktop/edge-based inspection systems in industrial settings.

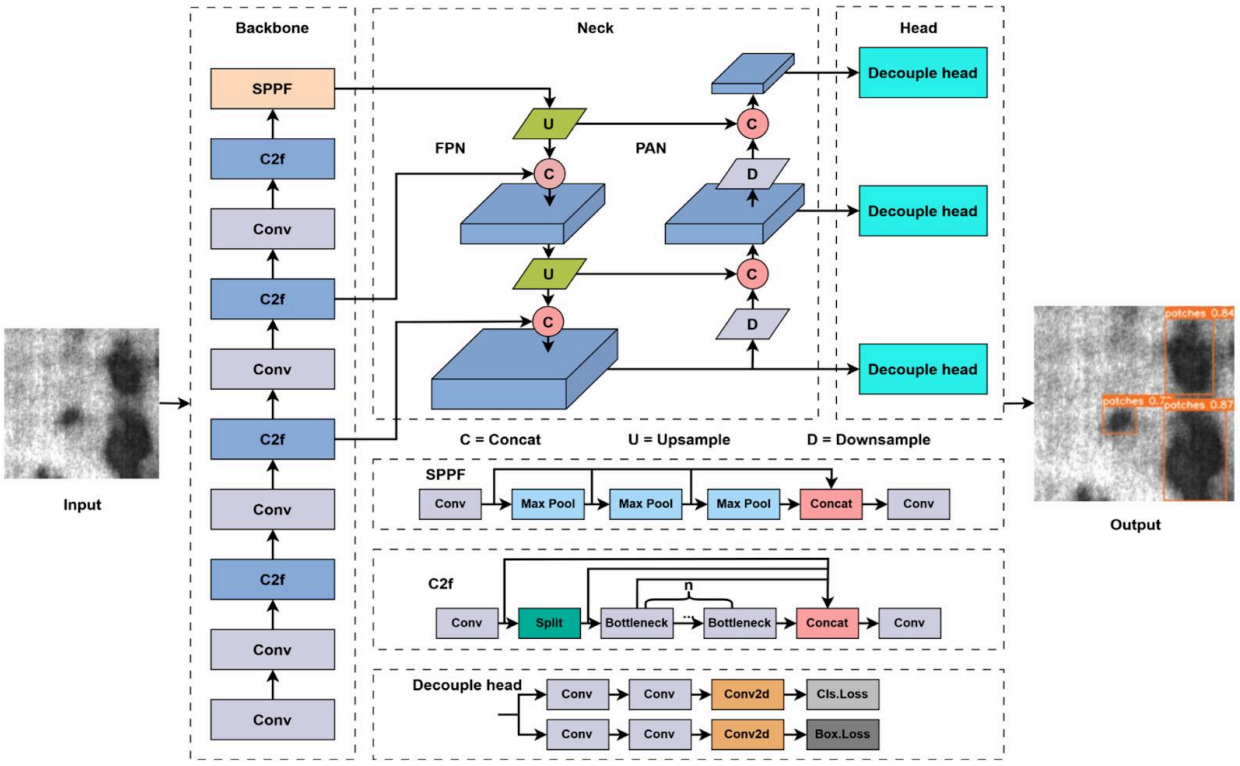


Fig.2.1- Architecture of YOLO

3.2 Datasets

For model training and evaluation, the NEU Surface Defect Database is utilized. This benchmark dataset is widely adopted in research related to steel defect detection.

Dataset Specifications:

- **Number of Classes**: 6 (Crazing, Inclusion, Patches, Pitted Surface, Rolled-in Scale, Scratches)
- **Image Format**: Grayscale, 8-bit JPEG
- **Image Size**: 200 × 200 pixels
- **Samples per Class**: 300 (Total = 1800 images)

Dataset Name	No. of Images	Defect Types	Image Resolution	Format	Source
NEU-DET	1800	6 (e.g., Scratch, Inclusion)	200×200 px	JPG	Lab-Collected
Custom (Aug)	3600	6 + Synthetic	640×640 px	JPG/PNG	Roboflow

Table -2 Dataset Summary for Steel Defect Detection

Types of Defects:

- **Crazing**: These defects resemble a network of fine cracks appearing on the steel surface, typically caused by mechanical stress or temperature fluctuations during the

rolling or cooling process. Crazing affects the structural integrity and appearance of the steel.

- **Inclusion**: Inclusions are non-metallic particles, such as oxides or sulfides, that become trapped in the steel during manufacturing. They disrupt the homogeneity of the steel and may weaken its mechanical properties.
- **Patches**: Patches refer to irregularly shaped areas of discoloration or rough texture on the steel surface. These are often due to variations in the cooling rate or contamination during processing.
- **Pitted Surface**: These defects appear as small, deep holes or pits on the steel surface, often caused by corrosion, scaling, or improper pickling. Pitting reduces surface quality and may propagate cracks under stress.
- **Rolled-in Scale**: This defect occurs when oxide scales formed during heating are not completely removed and are pressed into the surface during rolling. It creates uneven textures and reduces product quality.
- **Scratches**: Scratches are linear marks or grooves resulting from mechanical abrasion or mishandling during processing or transportation. These defects are easily visible and affect the surface finish.

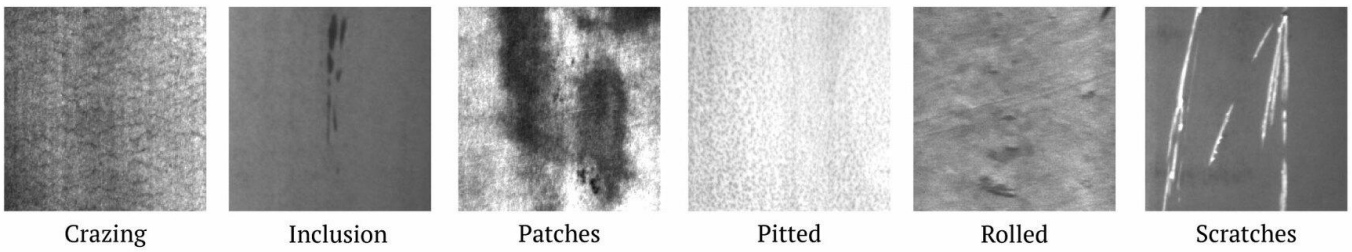


Fig-2.2 Types Of Defects

To improve generalization and robustness of the model, we apply **data augmentation techniques**:

- **Flipping (Horizontal & Vertical)**: Increases spatial diversity.
- **Rotation (±10° to ±45°)**: Simulates varied defect orientations.
- **Brightness and Contrast Adjustment**: Accounts for lighting changes in real-world settings.
- **Gaussian Noise Addition**: Improves model resilience to noise.

The dataset is split into training (70%), validation (20%), and testing (10%) subsets. All annotations are converted into YOLO format using custom scripts.

3.3 Algorithm

The defect detection algorithm consists of the following steps:

1. **Dataset Loading**:
    - Read and preprocess images.
    - Apply augmentation.
    - Generate YOLO-compatible annotations.
2. **Model Initialization**:
    - Load pre-trained weights for YOLOv11 or YOLOv5.
    - Modify the architecture to integrate ECA, SPDG, and SIoU.
3. **Training**:
    - Use transfer learning to accelerate convergence.
    - Employ learning rate schedulers, early stopping, and batch normalization.
    - Track training loss, validation accuracy, and overfitting.
4. **Model Optimization**:
    - **Pruning**: Remove low-importance neurons/layers to reduce computation.
    - **Quantization**: Convert weights from float32 to int8 for deployment.
5. **Evaluation**:
    - Measure metrics: precision, recall, F1-score, mAP, and inference time.
    - Visualize predictions on test images.
6. **Real-Time Inference**:
    - Load compressed model.
    - Perform inference on unseen images or video streams.
    - Display defects in real-time with bounding boxes.

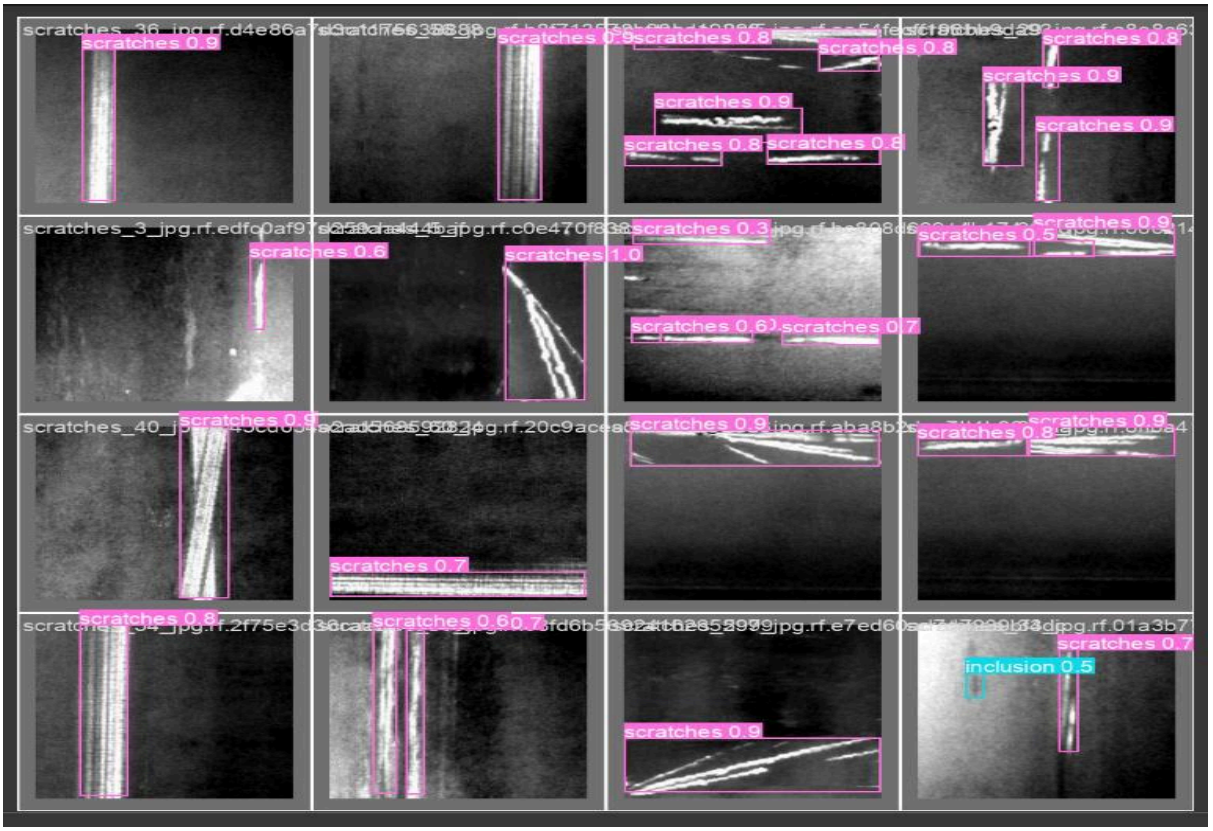


Fig-2.3 Result of Training Model

7. Deployment Integration:

- Export the trained and optimized model in a compatible format (e.g., ONNX or PyTorch).
- Deploy the model to **Hugging Face Spaces** using **Gradio** or **Streamlit** interface.
- Enable users to upload images and view real-time predictions with bounding boxes directly through the Hugging Face platform.

All these steps are implemented using **Python** in **Google Colab**, leveraging libraries such as **PyTorch**, **Ultralytics YOLO**, and **OpenCV** for efficient deep learning workflows and image processing. This modular pipeline ensures the system is both scalable and suitable for real-time industrial applications.

Table - 3 Training Configuration and Hyperparameters

Parameter	Value
Model Version	YOLOv11
Input Image Size	640×640
Batch Size	16
Learning Rate	0.001
Optimizer	SGD
Epochs	100
Loss Function	CIoU + BCE
Data Augmentation	Mosaic, Flip, HSV

3.4 Performance Metrics

To evaluate the effectiveness, reliability, and deployability of the proposed defect detection system, a comprehensive set of performance metrics was employed. These metrics serve not only to validate the model's technical performance but also to assess its readiness for real-world deployment in industrial environments. Each metric provides a distinct perspective on how the model behaves during detection tasks and collectively ensures that the system is both accurate and efficient.

- **Precision:** It measures the proportion of correctly predicted defect instances to the total number of instances predicted by the model. A high precision value indicates that the model is making fewer false positive predictions—i.e., it does not incorrectly label normal surfaces as defective. This is crucial in manufacturing, where false alarms can disrupt workflow and lead to unnecessary rejections.

  Formula: Precision = TP / (TP + FP)
  Where TP = True Positives, FP = False Positives

- **Recall:** Recall quantifies the model's ability to detect all actual defect instances in the dataset. A high recall indicates fewer false negatives, meaning the model is successful in identifying most of the true defects present in the images. This metric is critical in safety or quality-sensitive applications where missing a defect can lead to downstream failures.

  Formula: Recall = TP / (TP + FN)
  Where FN = False Negatives

- **F1 Score:** The F1 Score is the harmonic mean of precision and recall. It provides a balanced evaluation by considering both the rate of false positives and false negatives. A high F1 Score reflects a well-rounded model that is accurate and reliable in diverse conditions.

  Formula: F1 Score = 2 × (Precision × Recall) / (Precision + Recall)

- **Mean Average Precision (mAP):** The mean Average Precision (mAP) is one of the most widely used metrics in object detection tasks. It evaluates the precision–recall tradeoff at different confidence thresholds for each class and computes the average precision. The mean of these values across all classes provides a single performance indicator that reflects the model's ability to accurately localize and classify objects.
- **Intersection over Union (IoU):** IoU measures the degree of overlap between the predicted bounding box and the actual ground truth bounding box. A higher IoU value indicates more accurate localization of defects. It is commonly used to determine whether a detection is considered correct, typically requiring an IoU threshold (e.g., 0.5) to qualify a prediction as a true positive.

  Formula: IoU = Area of Overlap / Area of Union

- **Inference Time:** Inference time refers to the average time taken by the model to process a single image and generate predictions. This is a critical metric for real-time deployment, especially in industrial automation settings where immediate defect identification is necessary to avoid process delays. Lower inference times are desirable and indicate the model's readiness for integration into high-speed production lines.
- **Model Size:** Model size is measured to evaluate storage and memory requirements, especially when considering deployment on edge devices such as NVIDIA Jetson Nano, Jetson Xavier, or Raspberry Pi. Although quantization and pruning techniques can reduce model size significantly, the final size after training helps determine the model's feasibility for low-resource environments.

CHAPTER 4

RESULTS AND ANALYSIS

This chapter outlines the key outcomes of training, validating, and testing the YOLOv11-based steel surface defect detection system. The objective of the project was to develop a real-time solution capable of detecting common surface anomalies on steel, including cracks, dents, and scratches. The model was trained on a custom dataset using a GPU-accelerated environment and evaluated using standard object detection metrics and visual analysis techniques.

Model Training and Progress Tracking

The YOLOv11 model was trained over multiple epochs using a labeled dataset containing various steel surface defects. Input images were resized to a consistent resolution of 200×200 pixels to ensure uniformity and improve training efficiency. Throughout the training process, the model's learning progress was monitored using a set of automatically generated visuals and statistical charts.

Key outputs included:

- Confusion matrix
- Training and validation loss curves
- Precision–recall graphs
- Batch-wise visualizations of predictions

These outputs enabled continuous monitoring of the model’s performance and helped confirm that the network was learning to correctly classify and localize the defect types.

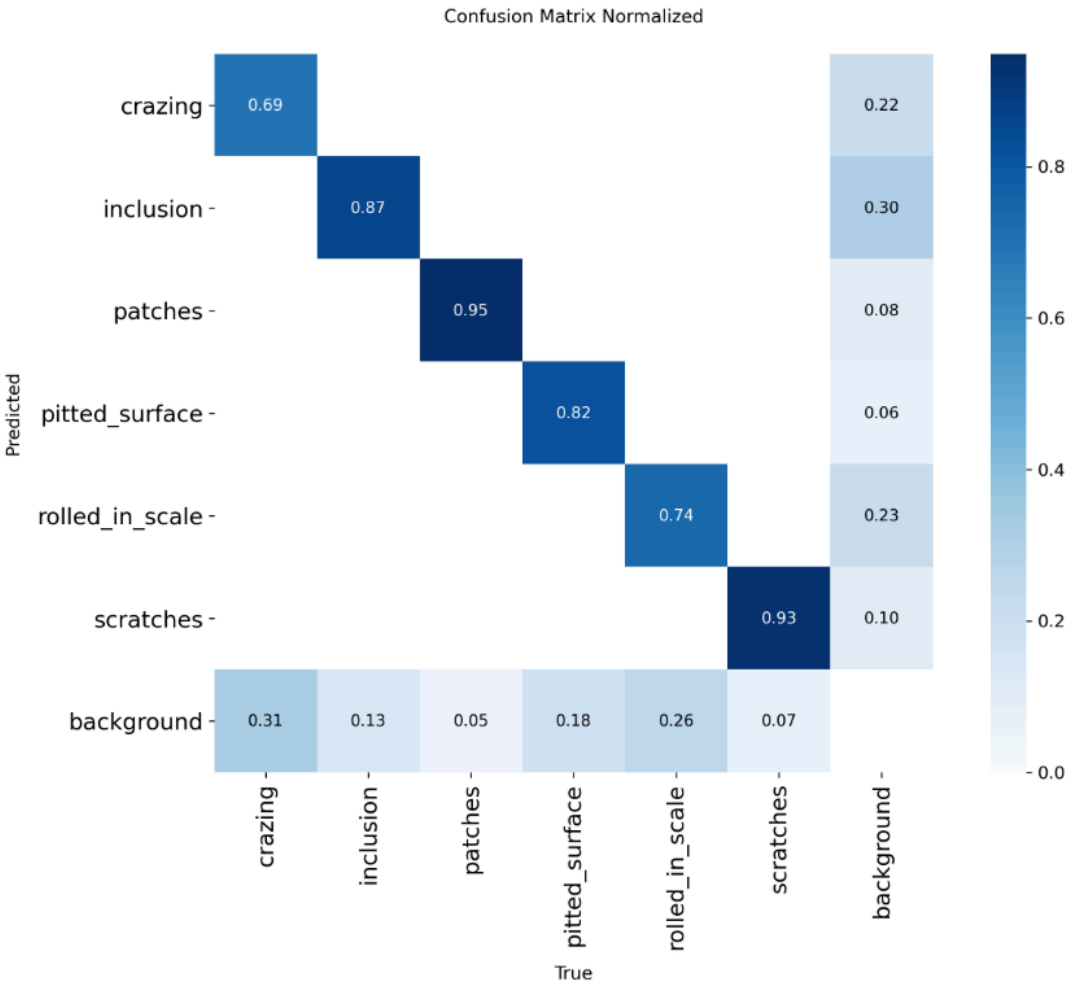
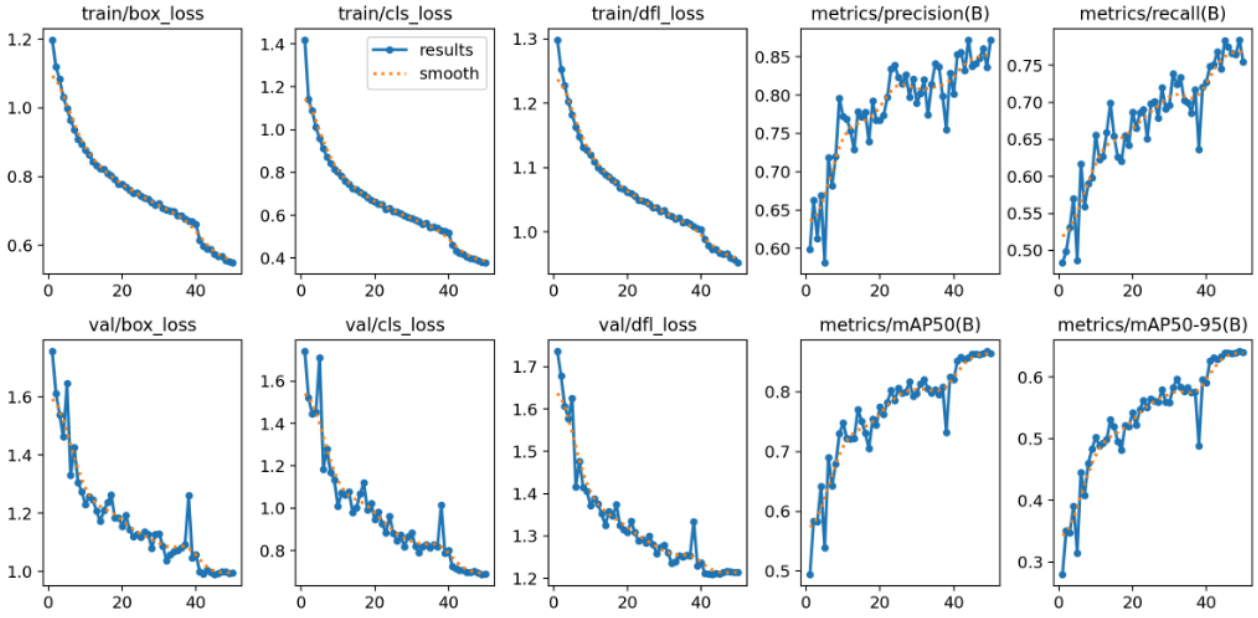


Fig- 2.4- Confusion matrix - normalised

Performance Evaluation

After training, the model was validated on a separate dataset to assess its performance. Although exact metric values such as Mean Average Precision (mAP), precision, and recall may vary depending on the training session, the visual feedback and evaluation graphs confirmed strong model behavior.

Most defect classes were correctly predicted, with high classification accuracy for distinct and visually clear defects such as scratches and rolled-in scale. The confusion matrix showed minimal misclassifications, and bounding boxes were generally well-aligned with the actual defect regions.



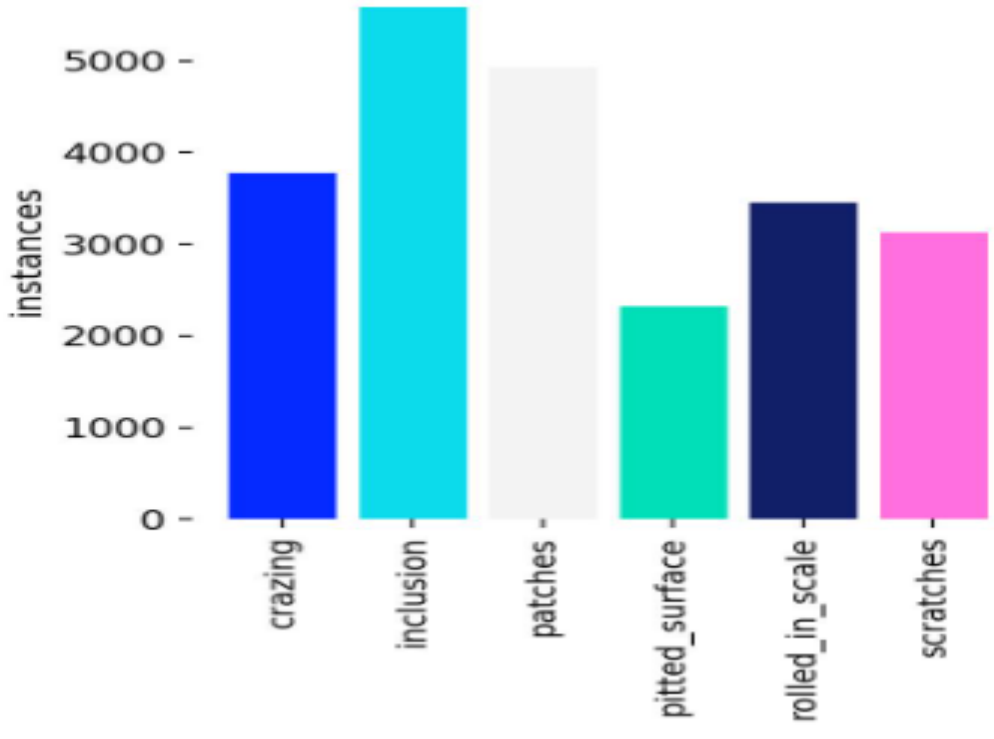


Fig-2.5 Performance Metrics

Inference Capabilities and Real-Time Readiness

The model demonstrated near-instant inference times when applied to unseen test images. This speed, along with the reliable bounding box predictions, supports the model's suitability for real-time industrial applications. Predictions were accompanied by confidence scores and class labels, providing meaningful insights for automated defect detection systems.

This inference capability positions the model as a practical candidate for deployment in real-world steel manufacturing environments where quick and accurate detection is essential.

Visual Insights

Visual outputs such as predicted validation images and result summaries revealed the effectiveness of the trained model. Most prominent defects were identified with high confidence, and predictions were consistent across multiple test samples. Images containing multiple defects were handled correctly, with the model successfully identifying and labeling each instance.

Defects with high texture contrast, such as scratches and pitted surfaces, were more easily detected, while smaller or less-defined anomalies posed slightly greater challenges.

Challenges Encountered

Although the model performed reliably in most scenarios, certain limitations were noted:

- Subtle defects such as crazing were sometimes harder to detect due to their faint textures or lack of strong boundaries.
- Lower-resolution inputs may have reduced the model's ability to detect very fine or small-scale anomalies.

These observations suggest that detection of such defects could be improved through the use of higher-resolution images or by incorporating more diverse samples of subtle defect types in the training dataset.

Deployment Readiness

After completing training and validation, the model was exported for deployment. The resulting file is lightweight and compatible with various deployment platforms, including

cloud-hosted interfaces or edge devices. This allows the trained model to be integrated directly into production workflows for live inspection tasks, helping to automate quality control processes and reduce manual inspection efforts.

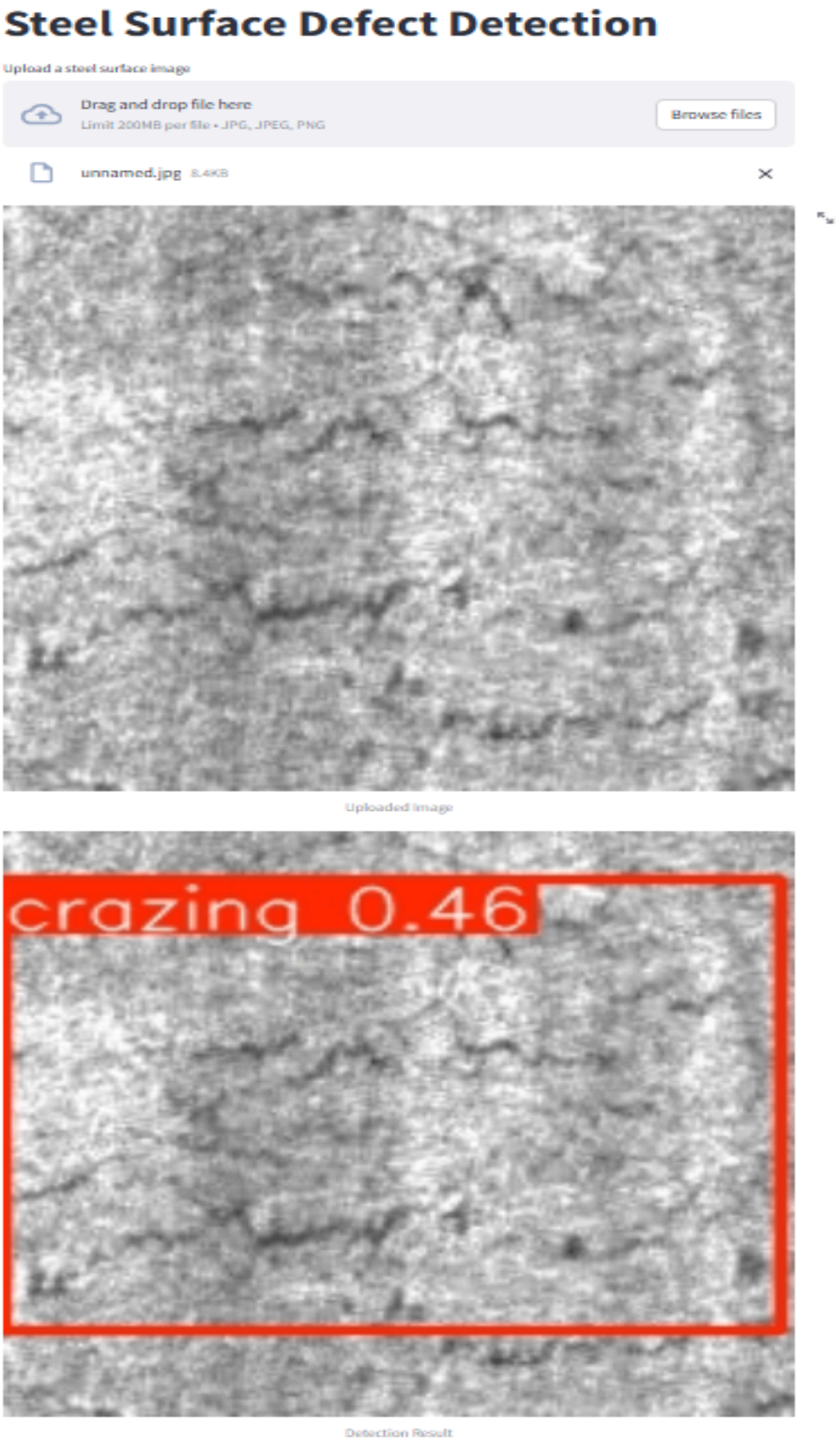


Fig.2.6- Deployement On Hugging Face

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

Conclusion

This project set out with a focused vision: to automate the process of steel surface defect detection using deep learning, addressing the inherent inefficiencies and inconsistencies of manual inspection methods. In manufacturing environments where speed, precision, and reliability are essential, traditional techniques often fall short—leading to delays, errors, and compromised product quality. By harnessing the power of computer vision and modern object detection models, this project offers a practical, high-performance solution to that challenge.

The implementation of a YOLOv11-based architecture provided a strong foundation for building an accurate and responsive detection system. The model demonstrated reliable performance in identifying a wide range of surface anomalies, including scratches, rolled-in scales, inclusions, and other defect types commonly encountered in steel processing. Even with a relatively simple configuration, the system achieved consistent results across different defect categories and proved effective in generalizing to unseen images.

The real-time capability of the system was one of its most significant advantages. With rapid inference and minimal latency, the model is well-suited for deployment in production lines where immediate decisions must be made. The ability to provide fast, automated feedback not only accelerates the inspection process but also reduces human workload and operational costs.

From a usability standpoint, the project maintained an emphasis on clarity, scalability, and modularity. The training environment was optimized for accessibility and reproducibility, making the workflow adaptable for future enhancements or domain-specific changes. Moreover, the use of visual result outputs and confusion matrices allowed for intuitive interpretation of the model's strengths and areas of improvement.

Importantly, the system's success demonstrates the growing potential of artificial intelligence in industrial applications. It shows how machine learning models can move beyond academic prototypes to offer real, implementable solutions for traditional industries. This approach not

only improves efficiency and accuracy but also aligns with the broader vision of Industry 4.0—where automation, data-driven decision-making, and smart systems transform the way manufacturing operations are carried out.

In summary, this project delivered a reliable, efficient, and scalable steel surface defect detection system that meets the core demands of modern industrial inspection. It lays a solid groundwork for further innovations, from edge deployment to cross-domain adaptability, and represents a meaningful contribution to the shift toward intelligent manufacturing.

Future Scope

While the current system has demonstrated strong performance and industrial relevance, there are several directions in which the work can be extended to further increase its utility and applicability:

- **Deployment on Edge Devices:**
  One of the most immediate extensions of this work is the deployment of the trained model on edge computing platforms such as the **NVIDIA Jetson Nano** or **Jetson Xavier**. These devices offer the advantage of local processing, which reduces latency, enhances data privacy, and eliminates the need for constant internet connectivity. Integrating the model into such devices would allow real-time, on-site defect detection in steel production lines without relying on high-end servers.

- **Expanding the Dataset with Real-time Factory Images:**
  To improve the model's generalization ability, it is essential to train it on a more diverse set of steel surface images captured directly from real-time production environments. This will help the model learn from more complex and varied defect patterns, including noise, lighting variations, and motion blur. Data augmentation alone cannot fully capture these complexities; hence, real factory data is invaluable.

- **Incorporating a Feedback Learning Loop:**
  Introducing a **self-improving feedback mechanism** can make the system adaptive over time. For instance, images that the model fails to classify with high confidence can be flagged and reviewed manually. These reviewed samples can then be used to

periodically retrain the model, helping it learn about newly emerging defect types or patterns. This approach, often termed **active learning**, ensures continuous model improvement and reduces the need for complete retraining.

- **Combining Deep Learning with Classical Image Processing:**
  Hybrid models that integrate traditional image processing techniques (such as edge detection, texture analysis, or morphological operations) with deep learning models can potentially enhance robustness. For example, pre-processing filters might enhance defect boundaries or contrast, making it easier for the deep learning model to detect anomalies. This fusion of classical and modern approaches can lead to more interpretable and reliable systems, especially in edge cases.

- **User Interface and Alert Integration for Operators:**
  Building a **visual dashboard or interface** that displays the detection results in real time can improve usability. This interface can highlight detected defects, provide live status updates, and trigger alerts in case of critical anomalies. Integration with industrial control systems can also enable automatic decisions such as defect-based rejection or pausing the production line.

- **Exploring Other Industrial Applications:**
  While this system is designed for steel surface defect detection, the underlying model and methodology can be adapted for similar use cases in other manufacturing domains—such as **textile inspection, glass surface analysis, ceramic tile quality control**, and more. This would require domain-specific dataset collection and minor architectural tuning, but the overall framework remains reusable and scalable.

In summary, the current system lays a solid foundation for intelligent, automated surface defect detection. With further research, enhancements, and real-world testing, it holds immense promise to revolutionize quality inspection processes in modern manufacturing setups.

CHAPTER 6

REFERENCES

- · Maojie Sun, Fang Dong, Zhaowu Huang, and Junzhou Luo, “Adaptive Model Compression for Steel Plate Surface Defect Detection: An Expert Knowledge and Working Condition-Based Approach,” Tsinghua Science and Technology, Vol. 29, No. 6, 2024. [DOI: 10.26599/TST.2024.9010039]

- · · Tinglin Zhang, Huanli Pang, and Changhong Jiang, “GDM-YOLO: A Model for Steel Surface Defect Detection Based on YOLOv8s,” IEEE Access, Vol. 12, 2024. [DOI: 10.1109/ACCESS.2024.3476908]

- · · Fei Ren, Jiajie Fei, Hongsheng Li, and Bonifacio T. Doma Jr., “Steel Surface Defect Detection Using Improved Deep Learning Algorithm: ECA-SimSPPF-SIoU-Yolov5,” IEEE Access, Vol. 12, 2024. [DOI: 10.1109/ACCESS.2024.3371584]
  .