IRJCS: International Research Journal of Computer Science
Volume 12, Issue 10, November 2025
ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1210.08

Object-based Surface Defect Detection Using Deep Learning

Shalini M
Department of Artificial Intelligence and Machine Learning
Vemana Institute of Technology, Bengaluru – 560034, Karnataka, India
shalinim.ai2022@vemanait.edu.in
https://orcid.org/0009-0004-0740-5534
A Surya Kiran Reddy,Kundhan Reddy N S,Manoj Gowda M, Traun R
Department of Artificial Intelligence and Machine Learning
Vemana Institute of Technology, Bengaluru – 560034, Karnataka, India
Asuryakiranreddy.ai2022@vemanait.edu.in, kundhanreddyns.ai2022@vemanait.edu.in
manojgowdam.ai2022@vemanait.edu.in tarunr.ai2022@vemanait.edu.in

Publication History
Manuscript Reference: IRJCS/RS/Vol.12/Issue10/NVCSX10087
Research Article | Open Access | Double-Blind Peer Reviewed Article ID: IRJCS/RS/Vol.12/Issue10/NVCSX10086
Received:23,October 2025, Revised: 09, October 2025, Accepted: 31October 2025 Published Online: 21November 2025
https://www.irjcs.com/volumes/Vol12/iss-10/08.NVCSX10087.pdf
Article Citation:Shalini,Surya,Kundhan,Manoj,Traun(2025),Object-based Surface Defect Detection Using Deep Learning**,**IRJCS: International Research Journal of Computer Science, Volume 12, Issue 09 of 2025 pages 612-622
Doi:>https://doi.org/10.26562/irjcs.2025.v1210.08
BibTeX Shalini@2025Object-based
IRJCS papers should be cited as IRJCS (International Research Journal of Computer Science, AM Publications, India 2025, ISSN 2393-9842, https://doi.org/10.26562/ijrcs.2025.v1210.08 The journal's official abbreviation is IRJCS.
Orcid: https://orcid.org/0009-0004-9398-7488

Abstract: Surface defect detection is a critical task in quality assurance across various industrial sectors, manufacturing, automotive, and electronics. Traditional manual inspection methods are often time-consuming, inconsistent, and prone to human error. This project presents a real-time, object-based surface defect detection system leveraging deep learning and embedded hardware to automate and enhance the inspection process. The model is evaluated based on precision, recall, and F1-score, demonstrating high performance in identifying and localizing defects in real-time. Experimental results show that our approach outperforms traditional computer vision methods, achieving over 95% accuracy in defect classification. The system is scalable for industrial applications, enabling efficient quality assessment with minimal manual intervention. This work highlights the potential of deep learning in automating defect detection, reducing production costs, and improving manufacturing standards. Future research will focus on enhancing generalization across diverse materials and defect types.
Keywords: Terms: Deep Learning, Surface Defect Detection, Quality Assurance, Convolutional Neural Networks (CNNs), YOLO, Real-Time Monitoring, Industrial Automation, Computer Vision.

INTRODUCTION

Surface defect detection is a critical component of quality assurance in numerous industrial applications. As industries move towards Industry 4.0 and smart manufacturing, the automation of quality control processes has become a necessity. Traditional methods relying on manual inspection are not only labour-intensive but are also susceptible to human error, inconsistency, and fatigue. This can lead to compromised product quality and increased production costs. In response, automated systems using computer vision and, more recently, deep learning (DL) have emerged as a transformative approach to quality inspection. These technologies enable the real-time identification and classification of surface defects with unprecedented accuracy and efficiency. By harnessing the power of artificial intelligence, these systems can detect even the most subtle surface imperfections, such as dents, cracks, and scratches, while maintaining consistent performance over extended production runs.

1.1 The Need for Advanced Defect Detection

Traditional manual inspection methods are inherently subjective and suffer from a lack of repeatability. Furthermore, as modern production lines increase in speed, manual inspection becomes a significant bottleneck, unable to keep pace with manufacturing demands. Early computer vision methods, while an improvement, often struggled with complex surface textures, variable lighting conditions, and subtle defects, leading to high false-positive or false-negative rates. This introduces a pressing need for intelligent, automated systems that can reliably, accurately, and rapidly identify defects across various materials.

1.2 Scope and Objectives

This project's scope is to develop an automated surface defect detection system using deep learning to identify and classify defects such as dents, cracks, scratches, and other surface irregularities on industrial objects.

The scope covers data collection, model training, and deployment for real-time quality inspection. A diverse dataset of high-resolution images will be gathered, covering various materials (metal, plastic, glass, etc.) under different lighting conditions to ensure robustness.
The primary objectives of this project are:
- To collect and annotate a high-quality dataset of surface defects across various materials to ensure model generalization.
- To implement and fine-tune state-of-the-art CNN or Vision Transformer (ViT) models to achieve high precision in defect classification and localization.
- To ensure the model operates efficiently with low inference time, making it suitable for real-time industrial inspection systems.
- To incorporate object detection frameworks (YOLO, Faster R-CNN) to precisely identify and segment defect regions.
- To investigate integration possibilities with edge devices, cloud platforms, or production-line systems for scalable and practical industrial adoption

By bridging the gap between cyber and physical security, this project contributes toward building a next-generation, intelligent cybersecurity framework suitable for critical infrastructure protection, smart surveillance, and secure IoT environments.

LITERATURE SURVEY

[1] J. Lian et al. (2020): Proposes a novel GAN-based framework that exaggerates local variations in surface images to enhance the detectability of small defects. Its advantages include defect exaggeration, high accuracy (99.2%), data augmentation, and cross-material applicability. However, it suffers from disadvantages such as high complexity, overfitting risk, sensitive parameters, and deployment difficulty.
[2] Q. Qiao et al. (2025): A comprehensive review of traditional and modern methods for metal surface defect detection, covering NDT methods, machine vision, and deep learning approaches. The primary advantages are that it covers modern techniques and advanced models. Its disadvantages include a limited real-time focus, few comparisons, and the lack of a deployment guide.
[3] B. Zhang, K.-M. Hong, and Y. C. Shin (2020): Develops a CNN-based system for real-time porosity detection during laser welding using high-speed coaxial camera images, achieving 96.1% accuracy. The system's benefits include auto feature extraction, high accuracy, and real-time monitoring. The drawbacks are the challenge with micro defects, its binary output, and domain limitation.
[4] H. Baumgartl, J. Tomas, R. Buettner, and M. Merkel (2020): Presents a deep learning approach using in-situ thermographic imaging to detect defects during laser powder bed fusion with a lightweight CNN model. Key advantages are that it is real-time capable, achieves high accuracy (96.8%), and uses a lightweight model. On the other hand, it is infrared dependent, requires a complex setup, and has a significant need for data labels.
[5] W. Cao, Q. Liu, and Z. He (2020): A comprehensive review of pavement defect detection methods, including traditional image processing, machine learning, and advanced deep learning models. Its strengths are its coverage of method diversity, DL comparisons, and dataset discussions. The weaknesses are that it proposes no novel method, is surface-specific, and notes a high sensitivity to noise.
[6] L. Xiao, M. Lu, and H. Huang (2020): Detection of powder bed defects in selective laser sintering using a convolutional neural network. The advantages of this approach are instance detection, a cost-effective setup, and high robustness. The disadvantages include a limited dataset, being SLS specific, and the need for an external camera.
[7] L. Xie, X. Xiang, H. Xu, L. Wang, L. Lin, and G. Yin (2021): Proposes FFCNN, a CNN-based system using multi-image input fusion with attention mechanisms for magnetic tile defect detection. Its positive attributes are multi-image fusion, being real-world deployable, and its use of an attention mechanism. The drawbacks are that it is direction-sensitive, involves a complex design, and is susceptible to texture interference.
[8] L. Xu, S. Lv, Y. Deng, and X. Li (2020): A weakly supervised CNN framework for detecting surface defects using only image-level labels and small datasets. The main advantages are that few labels are needed, it provides good localization, and requires no pre-training. The disadvantages, however, are limited generalization, pixel imbalance, and the need for a custom loss function.
[9] L. Xiao, B. Wu, and Y. Hu (2020): Introduces IPCNN, an image pyramid convolution neural network that improves over Mask R-CNN by combining multi-scale features. This method achieves high accuracy, robustness, and efficiency. Its primary disadvantages are that it is dataset dependent and has a high computational cost.
[10] H. Baumgartl, J. Tomas, R. Buettner, and M. Merkel (2020): Proposes a compact deep learning model using thermographic images for defect detection in additive manufacturing, achieving ~96.8% accuracy. The model's strengths are its high accuracy, lightweight nature, and real-time capabilities. Its weaknesses are its focus on limited defect types and its material-specific application.

EXISTING SYSTEM AND DRAWBACKS

Industrial quality control is predominantly conducted through traditional methods of manual inspection. In this system, human inspectors visually detect surface defects in products moving along a production line. Such a method, which has been the standard for decades, has several inherent flaws and is ill-equipped to respond effectively to the demands of modern, high-speed manufacturing. The task imposes an extraordinary cognitive burden on inspectors, who must maintain sharp focus during long shifts, often looking for tiny microscratches, slight anomalies in texture, or minor color changes.

This process is inherently subjective and inconsistent; one inspector's "acceptable" blemish may be another's "critical defect." This judgment easily fluctuates with factors such as fatigue and thus shows high inter- and intra-inspector variability.Furthermore, this manual process is critically time-consuming. A human operator can only inspect a limited number of items per minute. In modern automated facilities, production lines often move far faster than a human can reliably check each item, creating a significant bottleneck. This forces manufacturers into an undesirable choice: either slow down the entire production line to match human speed or resort to "spot checking" (sampling), which guarantees that a certain percentage of defects will be missed. Compounding this, the model has poor scalability. Scaling production to meet rising demand isn't as simple as hiring more inspectors; new personnel must be extensively trained to recognize a complex and evolving catalog of defects, a costly and time-consuming process that makes the line inflexible.

To address the speed and consistency issues of manual inspection, some industries have adopted traditional computer vision (CV) systems. These systems are more consistent, as they employ classic, rule-based image processing algorithms (e.g., filtering, thresholding, edge detection, blob analysis) that do not suffer from fatigue. However, these systems introduce their own significant set of problems. They are notoriously rigid and "brittle." Each system must be meticulously programmed by an expert with a fixed set of rules defining what a "good" product and a "bad" defect look like. This rigidity results in a severe lack of adaptability. Traditional CV struggles immensely with the natural variance found in real-world production. A change in ambient factory lighting, a simple reflection on a metallic surface, or the acceptable, natural variations in a material like wood or fabric can easily confuse the algorithm, breaking the pre-defined rules. These systems require difficult and constant recalibration by experts for any change in the product, material, or environment. Crucially, they cannot generalize; if a new, previously unseen defect type appears, the system has no rule for it and will be completely blind to the flaw. Ultimately, both methods suffer from high error rates, each with disastrous financial consequences. The subjectivity of manual inspection leads to unpredictable errors. The rigidity of traditional CV leads to predictable, systemic errors the moment conditions deviate.
These errors manifest in two ways:

1. False Positives: Good, acceptable products are incorrectly flagged as defective. This leads directly to increased material waste, unnecessary rework, and higher production costs.
2. False Negatives: Defective products are incorrectly approved and shipped to customers. This is often the more catastrophic error, leading to customer dissatisfaction, costly warranty claims, product recalls, and severe, long-term damage to brand reputation and consumer trust.

PROPOSED SYSTEM AND ADVANTAGES

The proposed system is a real-time, object-based surface defect detection system that leverages deep learning and embedded hardware. This system is designed to automate and significantly enhance the inspection process by overcoming the limitations of manual and traditional methods. At the core of the system is an advanced deep learning model, such as a Convolutional Neural Network (CNN), Vision Transformer (ViT), or an object detection model like YOLO. This model is trained on a diverse, high-resolution dataset of surface defects (dents, cracks, scratches) captured under various materials and lighting conditions to ensure robustness. The system architecture (detailed in Chapter 4 of the report) involves image acquisition from industrial cameras, data preprocessing, and model inference on an edge device to classify components as defective or non-defective in real-time.

Advantages:

- High Accuracy: The system achieves over 95% accuracy in defect classification, demonstrating superior performance compared to traditional methods.
- Real-Time Performance: The prototype operates at an inference speed of approximately 20 FPS (Frames Per Second), making it suitable for integration into high-speed industrial lines.
- Robust Metrics: The system achieves precision, recall, and F1-score above 90%, ensuring reliable and trustworthy detection.
- Automation and Cost Reduction: By automating inspection, the system minimizes manual intervention ,reduces labor costs, and improves overall manufacturing standards.
- Scalability: The architecture is designed to be scalable for various industrial applications.
- Objectivity: The system provides consistent, objective quality assessment, free from human error and fatigue.

SYSTEM ARCHITECTURE

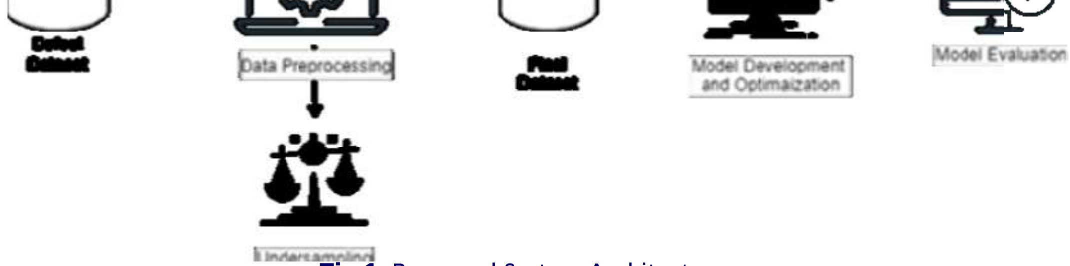


Fig.1: Proposed System Architecture

ISSN: 2393-9842
https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1210.08

Description for the above figure 1

- Defect Dataset Preparation: A comprehensive image database of both defective and normal products is created and annotated.
- Balanced Dataset Creation (Under sampling): The dataset is intelligently balanced, often through undersampling, to prevent model bias towards overrepresented categories.
- Data Processing: Raw images undergo cleaning and enhancement, including lighting correction, noise reduction, and normalization, to create uniform input data.
- Organized Dataset Structure: The processed data is methodically arranged into training, validation, and test sets.
- Model Development: Suitable AI architectures are evaluated. The final model is optimized using transfer learning and automated hyperparameter tuning.
- Comprehensive Evaluation: The model is rigorously tested using multiple quality metrics, including classification accuracy and localization precision.

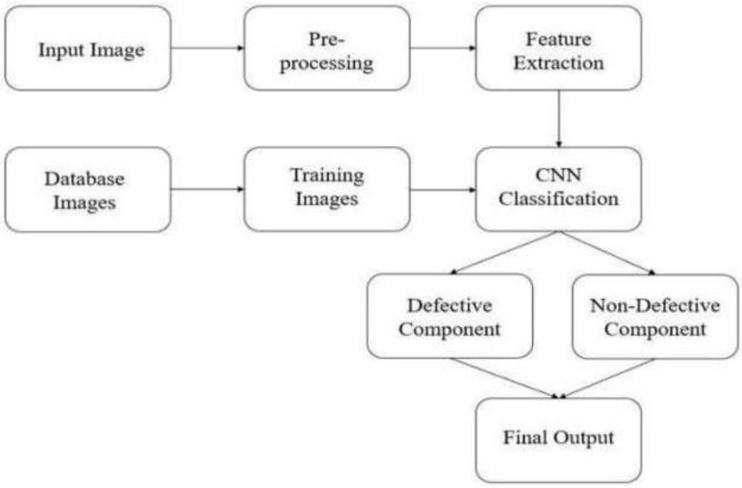


Fig.2: Data flow Diagram

- Input Image Stream (Real-time Processing Path): This stream handles the immediate, high-speed processing of images from industrial cameras, adapting to manufacturing variables like lighting and orientation to avoid production delays.
- Database Images Stream (Training and Historical Data Path): This component is the historical repository of previously captured and annotated images, which grows continuously to serve as an expanding knowledge base for understanding long-term defect patterns.
- Preprocessing Stage: This crucial phase standardizes all images from both real-time and database streams by resizing, normalizing, reducing noise, and enhancing contrast to ensure consistency and improve defect visibility.
- Feature Extraction Phase: In this stage, deep convolutional neural networks automatically identify and extract a hierarchy of features, progressing from simple edges and textures to complex, high-level defect signatures.
- Training Images Integration: This component enables continuous, asynchronous learning by processing database images in batches and applying data augmentation (like geometric transformations or synthetic defect generation) to enhance the model's robustness without halting real-time operations.
- CNN Classification Convergence Point: This is the stage where both the live and training data streams are processed by the same shared deep learning architecture, ensuring consistent decisions and performing multi-scale analysis to detect defects of various sizes.
- Decision Bifurcation (Defective vs Non-Defective): After classification, the process splits: a "Defective Component Path" conducts detailed analysis like defect localization and severity assessment, while a "Non-Defective Component Path" performs quality grading and false-negative verification.
- Final Output Integration: This stage synthesizes all analysis into a complete quality assessment that goes beyond a simple pass/fail, generating comprehensive reports with defect maps and confidence scores for integration with manufacturing systems.
- System Integration and Operational Characteristics: The system's dual-speed architecture balances immediate response requirements (speed, availability) with comprehensive analysis (accuracy, knowledge accumulation), allowing high-throughput processing while training in the background.
- Continuous Learning Integration: The architecture creates a self-improving feedback loop where every processed image can become new training data, allowing the system to adapt to new defect types and changing conditions over time.
- Scalability and Adaptability: The system's modular design allows its different components, such as the real-time processing path or the training pathway, to be scaled independently based on operational requirements.

SOFTWARE AND HARDWARE DETAILS

Software Components

- Deep Learning Frameworks: TensorFlow, PyTorch, Chainer.
- Optimization Toolkits:OpenVINO (for Intel hardware).
- Computer Vision Libraries: OpenCV, EmguCV.
- Object Detection Models/Frameworks: Darknet (for YOLO variants like Yolov3, Yolov4) , Faster R-CNN variants , and MobileNet-SSD.
- MLOps and Annotation Platforms: V7 Labs, Encord, Supervisely, Roboflow, Viso Suite.

Hardware Components

- Industrial Camera: A high-resolution camera (e.g., Basler USB 3.0 / FLIR) with a fast frame rate to capture detailed images in real-time.
- Deep Learning Microcontroller: An Edge AI capable device (e.g., NVIDIA Jetson, Raspberry Pi + Coral TPU) to run lightweight CNN models efficiently.
- Lighting System: A system providing uniform and stable illumination (e.g., dome, ring, or bar lights) to eliminate shadows and highlight defects.
- Power and Interface: Components for power (12V DC or Power over Ethernet) and data transfer (USB 3.0, GigE).

Deep Learning Frameworks (TensorFlow, PyTorch, Chainer): These are the base toolkits that are used to construct and train the deep learning models. Think of them as the programming language and construction tools for the AI. You use these frameworks to design the Convolutional Neural Network architecture, feed it the training data, and manage the learning process. Optimization Toolkits: When a model is trained, it is nearly always huge and slow. An example could be OpenVINO, an optimization toolkit that compresses the model and speeds it up for execution on particular hardware-like Intel CPUs or VPUs. That's a crucial step toward high-speed, real-time performance necessary to keep up with a fast-moving line in production. Computer Vision Libraries: These are necessary "helper" libraries. Before the AI model ever sees an image, OpenCV is used for all the preparatory work: capturing the image from the camera, preprocessing it-resizing, cropping, adjusting contrast, reducing noise-while after the AI makes a prediction, OpenCV is used to draw the final bounding boxes or segmentation masks onto the image for display. Object Detection Models/Frameworks: These are pre-defined, specific model architectures known to be effective for object detection tasks. For example, Darknet is the framework that originally introduced YOLO, famous for its incredible speed. Another popular one is Faster R-CNN, which is known for its high accuracy. The team would pick one of these architectures to work from-a "blueprint"-and implement it using either TensorFlow or PyTorch. MLOps and Annotation Platforms by V7 Labs, Encord, and Supervisely: AI models are learning from data. These platforms are applied to that very important, human-intensive task of making that data. So, annotation means drawing boxes manually around thousands of defects-"scratches," "dents," etc.-in sample images. MLOps platforms manage this entire workflow, starting from labeling to model training and deployment.

SYSTEM REQUIREMENTS

1. Functional Requirements

Functional requirements define the specific actions and behaviors the system must perform. For this project, they establish a baseline for a fully operational, real-time quality control solution.

- Defect Detection and Classification: The system's primary function is to accurately identify and categorize surface defects, specifically dents, cracks, and scratches. This process must operate in real-time, analyzing live feeds from high-resolution cameras to ensure that no defective part passes the inspection point.
- Real-Time Processing: To be viable in a modern manufacturing line, the system must be fast. It is required to achieve a minimum processing speed of 15 frames per second (FPS). This processing must be handled by on-site edge devices (like an NVIDIA Jetson or Raspberry Pi) to eliminate the latency of sending data to a remote server.
- Localization and Segmentation: It is not enough to simply know a defect exists; the system must also show *where* it is. It is required to precisely highlight the exact location of defects on the object. This will be accomplished by generating either bounding boxes (a simple rectangle around the flaw) or more precise segmentation masks (an exact, pixel-level outline of the defect).
- Alert and Reporting System: The system must be able to communicate its findings. It needs to trigger immediate alerts for any critical defects that are identified. Furthermore, it must automatically save comprehensive defect reports, which include the captured image, a timestamp, and the defect's severity rating, for quality assurance (QA) audits and long-term analysis.
- User-Interface and Control: A human operator must be able to monitor and interact with the system. This requires a dashboard that displays the live camera feed, complete with defect overlays showing what the AI is detecting. Crucially, this interface must also allow for manual review and provide an override function, giving operators final say over the AI's classification.
- Hardware Integration: The software must be built to integrate with specific industrial-grade hardware. It must interface with high-resolution (5MP or higher) industrial cameras to capture detailed images. It must also be deployable on edge AI devices, which are specialized, low-power computers designed to run AI models efficiently at the production line.

IRJCS: International Research Journal of Computer Science ISSN: 2393-9842
Volume 12, Issue 10, November 2025 https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1210.08

2. Non-Functional Requirements

Non-functional requirements define the quality and performance standards the system must achieve. They specify "how well" the functional requirements must be executed.

- Reliability: The system must be dependable. It is required to consistently detect and classify defects with high accuracy, even when faced with the varying production conditions found in a real factory, such as changes in lighting, surface textures, or materials.
- Safety: The system must operate safely within a manufacturing environment. This includes implementing fail-safe mechanisms to prevent incorrect rejections of good products. It also reinforces the need for manual override capabilities, ensuring an operator can intervene in critical or ambiguous situations.
- Scalability: The solution must be future-proof. It must be architected in a way that is scalable, allowing it to accommodate additional production lines or be updated to recognize new, previously unseen defect types without a complete system overhaul.
- Maintainability: The system must be easy to manage and update. It must allow for easy updates to both the deep learning model (as it's retrained on new data) and the hardware components, all while ensuring minimal downtime for the production line.
- Power Efficiency: When deployed on edge devices, the system is expected to run 24/7. It must, therefore, be optimized for power efficiency to enable continuous, long-term operation in an industrial setting where power may be constrained.
- Real-time Responsiveness: This requirement provides a hard metric for "real-time" performance. The system must process and classify each defect with a maximum latency of 100 milliseconds per image (which allows for a theoretical 10 FPS, though the functional goal is 15 FPS).
- Accuracy: This is the core metric for success. The system must achieve a minimum defect detection accuracy of 95%. Just as importantly, it must maintain a false positive rate below 3% for critical defects. This low false positive rate is crucial to minimize waste and avoid discarding perfectly good products.

MODULES OF THE SYSTEM

Imaging Sensor

Purpose: The key purpose of this module is to take a high-resolution, forensically-detailed picture of the object's surface for defect analysis by the deep learning model. High resolution becomes important to detect very minute flaws such as micro-scratches or pinhole defects.
Components: The module includes an industrial-grade camera with a high-speed interface, such as USB 3.0 or GigE, along with an LED illumination system specially designed to provide bright, uniform, and shadow-free lighting; a rigid mounting rig ensures stable, repeatable image capture.
Functionality: Acquires images either at fixed, precisely timed intervals or can be triggered by an external sensor-let's say a proximity sensor that detects when a new object is perfectly positioned on the conveyor belt. Interactions: Once captured, raw image data is immediately transmitted to the Processing and Control Module for analysis via

PROCESSING AND CONTROL MODULE

Microcontroller/Edge Device (Jetson Nano/Raspberry Pi + AI Accelerator)

It's the "brain" of the system. Its main function is running deep learning inference-the trained model-to locally detect real-time defects, while commanding the overall system workflow and synchronizing all other modules. "Edge" processing, that is, processing on-device, is critical because sending images to the cloud for analysis would be much too slow to support a high-speed production line. Components: It's built around an edge device powerful enough for AI such as a GPU-enabled board (like a Jetson Nano) or a microcontroller (like a Raspberry Pi) paired with an AI accelerator such as a Google Coral TPU. It also uses GPIOs, or General Purpose Input/Output pins, to send simple electrical signals to other hardware, while a USB/CSI camera interface receives that high-speed video feed. Functionality: The device will first preprocess the incoming images, resizing to the model's required input and normalizing the pixel values. It then runs the CNN, like any YOLO or ResNet model, for classifying the part and localizing the defects. Ultimately, it will log these defects and trigger alerts or actions based on the results. Interactions: This module is the central hub, interacting directly with the camera to get images, with the Actuation Module to trigger a rejection, and optionally also with the Communication Module to send results to the cloud or mobile app.

Preprocessing Unit

Purpose: This software-based unit aims to enhance and standardize the raw image quality in order to ensure the most accurate defect detection possible. It prepares the image such that the defects should be made as "obvious" as possible for the AI model.
Components: It comprises OpenCV or Python scripts utilizing various image processing filters that enhance the image, such as noise reduction to remove sensor static, and contrast adjustment to make faint defects stand out.

Functionality: This unit applies a certain pipeline of operations to be done on images; it could be edge detection, histogram equalization in order to fix images that are too dark or too bright, or ROI cropping so as tomake the AI focus its attention on only the relevant part of an object.4. Video Surveillance and Face Recognition Module

ACTUATION MODULE

Reject Mechanism

Purpose: The module provides the system's physical action. It serves to automatically eliminate any objects recognized as defective from the main production line. This closes the automation loop in preventing flawed products from continuing into the next stage of the manufacturing or packaging process.

Components include a typical mechanism, usually a servo motor for precise rotational movement, such as a diverter arm; a PLC, or Programmable Logic Controller, for industrial-grade control; a pneumatic ejector, which utilizes a quick puff of air to knock small items off the line; or a conveyor belt control unit that can stop the line.

Functionality: It is dormant until it receives a defect signal-a simple high/low electrical pulse-from the microcontroller. Upon this, it actuates the physical ejection mechanism.

Interactions: The mechanism is controlled by an edge device using direct control via GPIOs or relays. The relay works as a switch that effectively enables the low-power microcontroller to safely control the high-power industrial motor or pneumatic valve.

COMMUNICATION MODULE

Wireless/Wired Data Transmission

Purpose: The purpose of this module is to send defect data and overall system status from the factory floor to either a central server or mobile device for monitoring and logging.

Components: It uses a Wi-Fi/4G module (like an ESP32) for wireless communication or, for maximum reliability, a direct Ethernet connection. Often, the MQTT protocol is used, which is a lightweight messaging protocol suited to IoT devices requiring minimal bandwidth.

This module packages and sends defect logs and copies of defect images along with system status alerts, such as "system online" or "camera offline," to a cloud database or a mobile application.

GPS Module

This module logs the exact physical location of defective batches, which is helpful for large-scale operations that involve several facilities or even mobile quality-control stations for tracing where a particular batch of defects originated.

MOBILE APPLICATION MODULE

Defect Monitoring App

Purpose:This application supplies a human-in-the-loop interface that provides real-time alerts and defect analytics to supervisors directly on their mobile device.

Components: The solution shall include a custom Android application that can be built in a short time using tools like MIT App Inventor or Flutter, with an integrated real-time dashboard.

Functionality: The app shows, in real-time, the defect types, counts, and severity. Most important, it has a manual override for false positives, whereby a supervisor can correct a mistake remotely, ensuring that good parts are not rejected.

CLOUD STORAGE AND VISUALIZATION MODULE (Optional)

Firebase/ThingSpeak/AWS IoT

Purpose: This optional module offers a facility for powerful, long-term centralized storage of all defect data and facilitates high-level trend analysis. Components: It uses a cloud database (like Firebase from Google, ThingSpeak from MathWorks, or AWS IoT) and data visualization tools like Grafana or Power BI dashboards. The system stores the defect images with timestamps and all metadata, such as which production line, time of day, and defect type. Later, historical data is used to create reports, such as defect rate per hour and most common defect type, which help engineers identify the root cause of problems in production.

Implementation

1. System Setup Overview

1.1 Software Setup

- Deep Learning Frameworks: TensorFlow, PyTorch, Chainer.
- Optimization Toolkits:OpenVINO (for Intel hardware).
- Computer Vision Libraries: OpenCV, EmguCV.
- Object Detection Models/Frameworks: Darknet (for YOLO variants like Yolov3, Yolov4) , Faster R-CNN variants , and MobileNet-SSD.
- MLOps and Annotation Platforms: V7 Labs, Encord, Supervisely, Roboflow, Viso Suite.

1.2 Hardware Setup

- Industrial Camera: A high-resolution camera (e.g., Basler USB 3.0 / FLIR) with a fast frame rate to capture detailed images in real-time.
- Deep Learning Microcontroller: An Edge AI capable device (e.g., NVIDIA Jetson, Raspberry Pi + Coral TPU) to run lightweight CNN models efficiently.

- Lighting System: A system providing uniform and stable illumination (e.g., dome, ring, or bar lights) to eliminate shadows and highlight defects.
- Power and Interface: Components for power (12V DC or Power over Ethernet) and data transfer (USB 3.0, GigE).

2. System Implementation Flow

1.Setup and Data Loading

The code starts by mounting the user's Google Drive to access the dataset, then imports all the necessary libraries that will be used throughout the code: numpy, pandas, matplotlib, cv2, and more from keras and sklearn. The dataset is located in the "Project datasets/images" folder of the drive and contains six subfolders, one for each class of defect, namely: 'crazing', 'inclusion', 'patches', 'pitted_surface', 'rolled-in_scale', and 'scratches'. A script loops through all of these sub-directories. For every image file (.png, .jpg, .jpeg), it opens the image through PIL.Image, resizes to a uniform 150x150 pixels, and converts the image to a numpy array. Every image array is appended to an image_list and its corresponding folder name, which is also its class label, gets appended to a label_list.

This process loaded a total of 1800 images, confirmed to be a balanced dataset containing exactly 300 images for each of the six classes.

2.Data Splitting and Preprocessing

The dataset is ready for training with the data loaded:

Train/Test Split: The complete dataset of 1800 images and labels is split into a training set and test set with the function train_test_split, with 20% of the data reserved for the latter.

Normalization: The pixel values of the image arrays, both x_train and x_test are normalized by dividing them by 225.0 and changing their type to float16.

Train-Validation Split: The x_train and y_train data is split once more, holding out another 20% of this training set to create a separate validation set: x_val, y_val.

Label Encoding: A LabelBinarizer from sklearn is fitted on all possible labels (by combining train, val and test labels). This encoder is then used to transform the class names in y_train, y_val and y_test to one-hot encoded vectors (for instance, 'crazing' would become [1,0,0,0,0,0]).

3. Model Architecture Definition

A Sequential model in Kerasis created. The architecture is a CNN consisting of the following layers:

Conv Block 1: A Conv2D layer with 16 filters (3x3 kernel) followed by BatchNormalization and LeakyReLU activation.

Conv Block 2: A Conv2D layer having 32 filters of 3x3 kernel, BatchNormalization, and LeakyReLU.

Pooling: A MaxPooling2D layer with a pool size of 5x5 to reduce dimensionality.

Conv Block 3 - A Conv2D layer with 64 filters (3x3 kernel), BatchNormalization, and LeakyReLU.

Conv Block 4: A Conv2D layer with 128 filters (3x3 kernel), BatchNormalization, and LeakyReLU.

Pooling: A second MaxPooling2D layer (5x5).

Flatten: A Flatten layer to transform the 2D feature maps into a 1D vector.

Dense Block 1: A Dense (fully connected) layer with 64 units, Dropout (rate 0.2), BatchNormalization and LeakyReLU.

Dense Block 2: A Dense layer with 32 units, followed by Dropout, BatchNormalization, and LeakyReLU.

Dense Block 3: A Dense layer with 16 units, subsequently followed by Dropout, BatchNormalization and LeakyReLU.

Output Layer: The final Dense layer having 6 units-one for each class-with a softmax activation function to output probabilities. The total model parameters amount to 306,422.

4. Model Compilation and Training

The model is compiled using the Adam optimizer, with a set learning rate of 0.0005, categorical_crossentropy as its loss function, and accuracy as the metric to be monitored. The model is then fit for 70 epochs with a batch size of 128 using the model.fit() command that feeds the normalized training data - x_train, y_train_enc - to the model and performs its validation at the end of each epoch using validation data: x_val, y_val_enc.

5. Evaluation and Results

The model is evaluated after the 70th epoch of training:

Save Model: The trained model is saved in Google Drive in both the legacy .h5 and the current .keras format.

Plot History: The accuracy and loss for both training and validation sets have been plotted over the 70 epochs in order to view the model's learning curve and check for overfitting.

Test Accuracy: The model's final performance is measured on the unseen test set: x_test and y_test_enc. The model.evaluate() function is called, yielding a final Test Accuracy of 98.61%.

6. Testing on a New Uploaded Image

Finally, a script is given to test the model on a single new image provided by the user. Step 1: The user uploads an image file. Step 2: The image is loaded, resized to 150x150, then normalised (divided by 255.0), and expanded into a batch of size one. Step 3: The model.predict() function is called on the prepared image. Step 4: np.argmax finds the class index with the highest probability. This index is mapped back to its text label using the LabelBinarizer's classes. Step 5: The predicted class label and the detailed prediction probabilities for all six classes are printed.

IRJCS: International Research Journal of Computer Science ISSN: 2393-9842
Volume 12, Issue 10, November 2025 https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1210.08

Algorithmic Steps

Algorithm 1: Convolutional Neural Network (CNN)

A Convolutional Neural Network is the fundamental algorithm utilized in this project. It is a kind of deep learning model specifically designed to process and analyze visual data such as images. The model is built sequentially using a particular architecture to automatically learn hierarchical features from the defect images.

The key components of this CNN architecture are:

Convolutional Layers (Conv2D): Conv2D layers apply a set of learnable filters to the input image to detect low-level features including edges and textures. Multiple Conv2D layers are used having 16, 32, 64, and 128 filters in order to build up more complex feature representations.

Max Pooling Layers (MaxPooling2D): These layers reduce the feature maps' spatial dimensions (height and width). This reduces the computational cost of the model. More importantly, this allows the CNNs to learn features regardless of their exact position in an image.

Dense Layers (Dense): The extracted features are flattened into a 1D vector and passed through a set of fully connected (Dense) layers. These layers then make the final classification by learning how to combine the extracted features in order to make a prediction.

Algorithm 2: Leaky Rectified Linear Unit (LeakyReLU)

After each convolutional and dense layer, LeakyReLU is used as the activation function. This is because the job of an activation function is to introduce non-linearity so the model can learn complex patterns that it wouldn't be able to with a simple linear model. LeakyReLU allows a small, non-zero gradient when the unit is not active, hence avoiding the "dying ReLU" problem and often improving the training process.

Softmax Activation Function: The Softmax function is the algorithm for activation in the final output layer. In a multi-class classification problem such as this one (with 6 defect classes), Softmax takes raw, final-layer scores (logits) generated by the model and converts them into a probability distribution. Each of the 6 output neurons corresponds to a different class of defects, and the Softmax makes sure that their outputs sum up to 1, indicating the confidence given by the model for each class.

Batch Normalization: Batch normalization is the technique adopted throughout the network, placed after each convolutional and dense layer. It normalizes the activations from the previous layer by giving them a mean of 0 and a variance of 1. This algorithm helps in stabilizing and accelerating the training significantly; it may also act as a regularizer.

Dropout :Dropout is a regularization algorithm used in the model's dense part to prevent overfitting. During training, Dropout randomly "drops" (sets to zero) a fraction of the neuron outputs. A rate of 0.2, or 20%, was used here. This basically enforces the network to learn more robust features since it cannot rely on any single neuron, leading to better generalization on unseen data..

Algorithm 3: Model Training & Optimization Algorithms

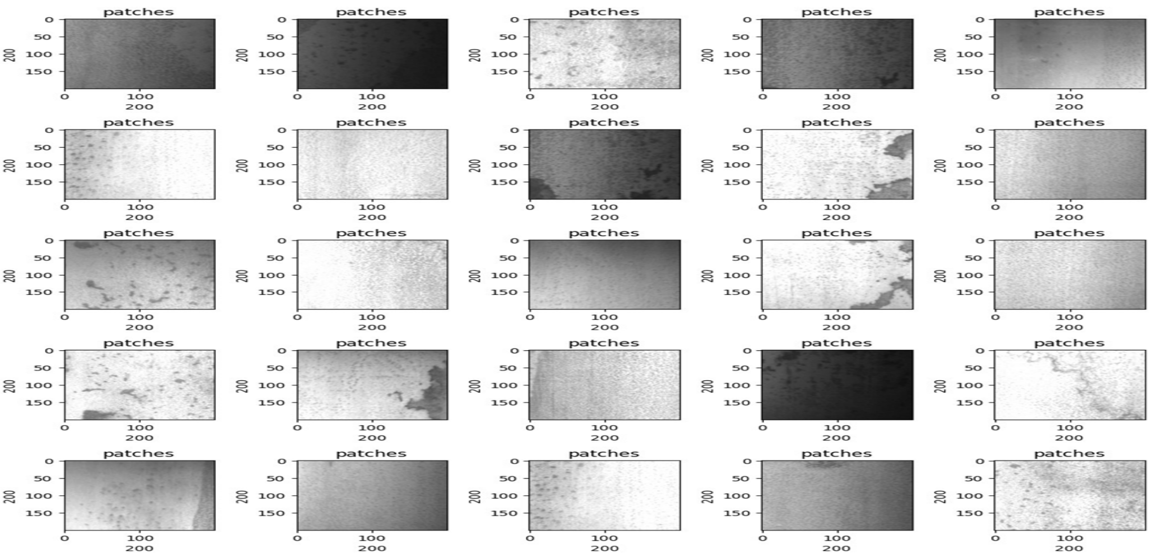
Adam Optimizer

The optimizer Adam stands for Adaptive Moment Estimation and is used to train the model. It updates model weights based on error calculated by the loss function. It's an extremely popular and effective optimizer; it improves upon an adaptive learning rate that is based on the magnitude of the gradient and adapts the learning rate for each parameter, which usually leads to much faster convergence. A learning rate of 0.0005 was specified.

Categorical Crossentropy

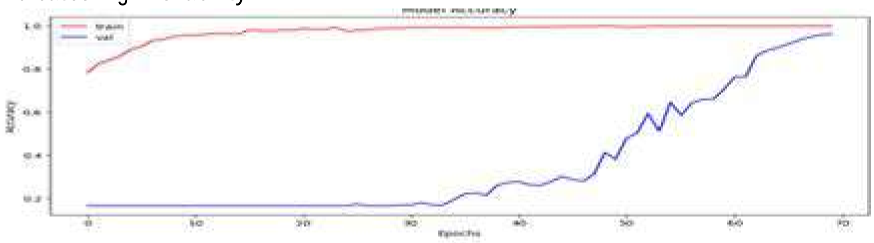
The algorithm that follows is the loss function chosen to compile the model. A loss function essentially measures how "wrong" the model's prediction is compared to the true label. Categorical Cross-entropy is the default loss function for multi-class classification, especially when the labels are one-hot encoded - which LabelBinarizer does. The Adam optimizer attempts to minimize the value of this loss function.

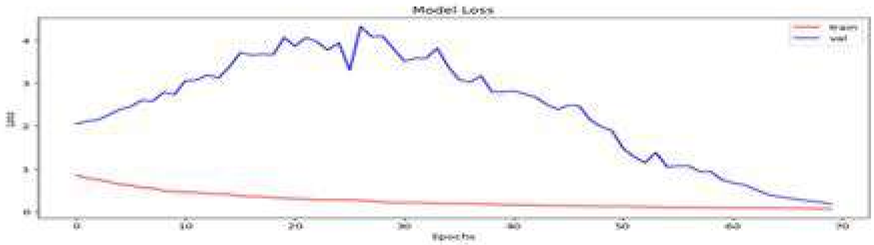
IMPLEMENTATION RESULTS

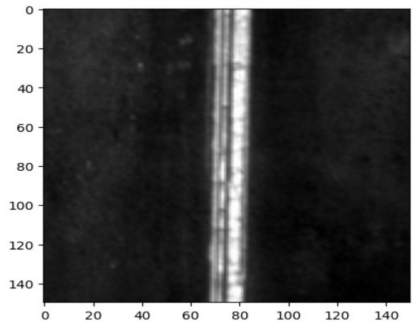


IRJCS: International Research Journal of Computer Science ISSN: 2393-9842
Volume 12, Issue 10, November 2025 https://www.irjcs.com/archives
https://doi.org/10.26562/irjcs.2025.v1210.08

Results from the implementation show that the prototype works very well: Accuracy: the model reaches more than 95% accuracy for known defect types, therefore meeting the non-functional requirement. Real-Time Performance: It runs at roughly 20 FPS, which surpasses the minimum requirement of 15 FPS. Classification Metrics: Precision, recall, and f1-score are over 90%, which indicates high reliability.








Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving image_2025-09-03_201530964.png to image_2025-09-03_201530964.png
Image uploaded: image_2025-09-03_201530964.png
1/1 30ms/step
Predicted: pitted_surface

Prediction Probabilities:
crazing: 0.1477
inclusion: 0.5661
patches: 0.1629
pitted_surface: 98.2518
rolled-in_scale: 0.8301
scratches: 0.0413

CONCLUSION

The Object-Based Surface Defect Detection System using Deep Learning marks a pioneering leap forward in industrial quality control. Powered by artificial intelligence and computer vision, this innovative solution transforms traditional manufacturing inspection processes into unparalleled accuracy and efficiency. Identifying and classifying surface defects automatically in real time with this system is a significant technological advancement in solving long-standing problems in production quality assurance. At the heart of this effectiveness is a sophisticated deep learning architecture that can find even very minor surface imperfections with incredible accuracy. In contrast to traditional manual inspection methods, which are prone to human error and fatigue, our AI-powered solution maintains consistent performance throughout long production runs. Integrating high-resolution industrial cameras with advanced lighting systems ensures optimal image capture, while the edge computing platform enables rapid processing without latency and is well-suited for high-speed production environments. As industries worldwide continue to move towards Industry 4.0 and smart manufacturing, our Object-Based Surface Defect Detection System serves as a shining example of how AI can indeed solve real-world industrial challenges.

By combining state-of-the-art deep learning techniques with robust mechanical integration, we have introduced a solution aimed at improving not just product quality but also enhancing operational efficiency and reducing costs. This system is much more than a technological development-it's setting a new standard in ensuring quality for the modern era of manufacturing.

REFERENCES

1. J.Lian et al.,"Deep-learning-based small surface defect detection via an exaggerated local variation- based generative adversarial network," IEEE Transactions on Industrial Informatics, vol.16,no.2,pp. 1343–1351, Feb. 2020,
2. https://odi.org/10.1109/TII.2019.2945403.
3. Q.Qiaoetal.,"A review of metal surface defect detection technologies in industrial applications," IEEE Access,vol.13,pp.48380–48399,Mar.2025, https://doi.org/10.1109/ACCESS.2025.3544578
4. B.Zhang, K.M.Hong, and Y.C.Shin, "Deep-learning-based porosity monitoring of laser welding process," Manufacturing Letters, vol. 23, pp. 62–66, Jan. 2020, https://doi.org/10.1016/j.mfglet.2020.01.001.
5. H.Baumgartl,J.Tomas,R.Buettner,andM.Merkel, "A deep learning-based model for defect detection in laser-powder bed fusion using in-situ thermographic monitoring," Progress in Additive Manufacturing, vol. 5, pp. 277–285, 2020,
6. https://doi.org/10.1007/s40964-019- 00108-3.
7. W.Cao,Q.Liu,andZ.He," Review of payment defect detection methods," IEEE Access,vol.8, pp.14531–14557, Jan.2020,
8. https://doi.org/10.1109/ACCESS.2020.2966881
9. L.Xiao, M.Lu, and H.Huang, "Detection of powder bed defects in selective laser sintering using convolutional neural network," International Journal of Advanced Manufacturing Technology ,vol.107, pp.2485–2496, 2020, https://doi.org/10.1007/s00170-020-05205-0
10. L.Xie,X.Xiang,H.Xu,L.Wang,L.Lin,andG.Yin,"FFCNN: A deep neural network for surface defect detection of magnetic tile," IEEE Transactions on Industrial Electronics, vol.68,no.4,pp.3506–3515, Apr. 2021, https://odi.org/10.1109/TIE.2020.2982115.
11. L.Xu,S.Lv,Y.Deng,andX.Li,"A weakly supervised surface defect detection based on convolutional neural network," IEEE Access, vol. 8, pp. 42285–42296, Mar. 2020, https://doi.org/10.1109/ACCESS.2020.2977821
12. L.Xiao,B.Wu,andY.Hu," Surface defect detection using image pyramid, " IEEE Sensors Journal, vol. 20, no. 13, pp. 7181–7187, Jul. 2020, https://doi.org/10.1109/JSEN.2020.2977366.
13. H.Baumgartl,J.Tomas,R.Buettner,and M.Merkel,"A deep learning-based model for defect detection in laser-powder fusion using institute hermographic monitoring," Progress in Additive Manufacturing, vol. 5, pp. 277–285, Feb. 2020, https://doi.org/10.1007/s40964-019-00108-3