Comparative Study of Deep Learning Algorithms for Real-Time Manufacturing Fault Detection

Shripad V. Kale[1], Prasad S. Laware[2], Mayank A. Agrawal[3], Nitin S. Khachane[4], Sohan R. Pachghare[5], Sunil R. Gupta[6]

[1,2,3,4,5,6]*Artificial Intelligence & Data Science Prof. Ram Meghe Institute of Technology & Research, Amravati,444701, Maharashtra, India*

***Abstract*—Quality control is a necessary part of the manufacturing process. Manual quality control is being applied in many small and medium enterprises (SMEs). Human observers are known to miss 20-30% of the defects due to fatigue and inconsistency. There are Visual Inspection (AVI) systems, which are automated and require expensive industrial equipment and special conditions of lighting. This renders them inapplicable to low budgets. The authors talk about the most recent algorithms that can be used to detect surface defects in this paper with the help of Deep Learning (DL) methods. Two-Stage detectors, e.g., Faster R-CNN, and Single-Stage detectors, e.g., YOLO, will be compared to each other. It is hoped to discover which solution is more feasible and effective when it comes to real-time low-budget apps with the use of standard webcams. The findings indicate that the Two-Stage detectors are not a lot more accurate than the Single-Stage detectors, but the Single-Stage detectors possess a sufficient frame rate (FPS).**

***Index Terms*—Defect Detection, Deep Learning, YOLO, Faster R-CNN, Quality Control, Computer Vision.**

I. INTRODUCTION

Automation of quality control is now a priority for the manufacturing industry, such as the automotive and electronics industry, with Industry 4.0. The conventional technique of quality control is very dependent on the human eye. There is, however, a limit to human visual inspection. As a result of numerous experiments, it is possible to prove that, due to a considerable duration of working hours, the accuracy of human eyesight inspection decreases by far and error occurs in one out of 10 to 20%. Moreover, the human eye inspection may also fail to be impartial until the quality standards of various shifts are not consistent.

In order to solve these issues, manufacturers have started the use of Automated Visual Inspection. Earlier AVI systems were based on traditional edge detection and thresholding image processing techniques. These are quick systems that are not very reliable. All it takes is a small alteration in the lighting of the factory or the positioning of the product, and it can be rendered ineffective.

Deep Learning (particularly, Convolutional Neural Networks) has transformed this domain entirely. Unlike the earlier methods, CNNs are able to acquire complex defect patterns-scratches, cracks or dents- in harsh environments. The only problem with students and small to medium enterprises is, however, to balance between cost and accuracy of computation.

The vast majority of the state-of-the-art works are directed to the achievement of maximum precision by the powerful GPU servers and industrial cameras. The algorithm implementation in the deployment perspective of low-budget hardware, such as laptops with normal webcams, is reviewed in this paper.

The existing Object Detection algorithms can be classified into two groups:

1. Two-Stage Detectors (e.g., Faster R-CNN): These propose regions and label them. They are highly precise but slow with inference rates of approximately less than 5 frames in non-GPU systems. Faster R-CNN is one example.
2. Single-Stage Detectors (e.g., YOLO, SSD): These consider the detection as a single regression problem that directly learns to predict the pixels of bounding boxes as absolute image positions. Such models are focused on speed and real-time utilisation. In this paper, these architectures have

been compared in such a way that the most appropriate one that is applicable in low-cost fault detection is established.

The following sections of this paper have the following structure: Section II presents the Literature Review, Section III provides a Comparative Analysis of the algorithms, and Section IV is a summary of the survey.

II. LITERATURE REVIEW

The field of defect detection has advanced significantly over the past ten years. This section looks at the shift from traditional image processing to modern Deep Learning architectures.

A. Traditional Image Processing Techniques: Traditional techniques involved the use of statistical and filtering methods. According to Ngan et al., regularity analysis came in useful to test textured patterns, which worked with homogenised products like textiles. Similarly, Leon used Gabor filters to identify structural damage in clothes through the comparison of the vibrations of frequencies in pictures. But the techniques cannot work well with noisy images. Song and Yan described that the conventional Local Binary Patterns algorithm is highly sensitive to pixel noise and is likely to confuse dust or grease as a crack.

B. Two-Stage and Heavyweight Models: Convolutional Neural Networks (CNNs) came into existence and were another significant milestone. As Tabernik et al. have shown, the networks that use the concept of segmentation like Mask R-CNN, could identify the exact shape of a defect. The Faster R-CNN proposed by Ren et al. reached the highest accuracy of 98.2% in the NEU-DET steel surface dataset [3]. These networks are very precise, however, they are also computationally demanding. Liu et al. reviewed various object detecting algorithms and discovered that the Two-Stage detectors tend to run at 5-7 FPS with average CPUs, which cannot fit into the production lines with high speed requiring immediate response due to speediness [4].

C. Single-Stage and Lightweight Models: To solve the issue of speed, Redmon and Farhadi [11] introduced the YOLO (You Only Look Once) model, which scans an image once. Newer attempts have been made to optimize such models for edge devices. YOLOv7, introduced by Wang et al. [12], utilised the so-called bag-of-freebies techniques without raising inference time to enhance the accuracy of the detection. MobileNet models have been considered in ultra-low-resource settings.

According to Howard et al. [13], depth-wise separable convolutions were capable of shrinking the model size by 90%. To satisfy our "low-cost" goal, Li [14] made use of a

tiny-YOLO model that could be executed in a Raspberry Pi and run snout defect detection at 15 FPS. However, Zhu et al. [5] found that lightweight models are quick, but not very efficient in identifying small objects. Recent models like YOLOv8 [6] are trying to make advances in order to eliminate this issue. A Survey by Zhou et al. [15] confirmed that Single-stage detectors are most ideal in compromising with SMEs in the industry.

III. CLASSIFICATION OF TECHNIQUES

In order to get a clearer picture of defect detection methods, we divided the existing solutions into two broad categories, including Traditional Computer Vision and Deep Learning.

A. Traditional Computer Vision: Before deep learning, the process of detecting defects relied on structural and statistical analysis.

- Statistical Methods: These methods, such as the histogram thresholding analyze the pixel intensities distribution. They are not heavy and cannot cope with complicated backgrounds.
- Filter-Based Methods: With such algorithms as Gabor filters and Canny Edge detection, the goal is to bring out the discontinuities of the surface texture. Simple shapes (such as cracks on a flat wall) work fine, but each new line of products needs to be tuned manually.

B. Deep Learning Approaches: Deep Learning does not require extracting features manually because it learns features directly from data. This can further be classified into:

- Supervised Learning: The model will undergo training based on labelled data, which is comprised of images with boxes around defects. This is the normal procedure of YOLO and R-CNN.

- Unsupervised Learning: The model determines the patterns that are normal and marks what seems to be abnormal. This can be useful where there is little such data on the defects, but is more complicated to apply in the case of undergraduate work.

Table I: Comparison of Recent Fault Detection Approaches				
Paper Ref.	Year	Algorithm Used	Hardware	Application
[1]	2024	Faster R-CNN	Industrial Camera	Steel Surface
[2]	2023	SVM + HOG	Standard Webcam	Bottle Caps
[3]	2025	YOLOv8	Laptop Webcam	PCB Defects

IV. COMPARATIVE ANALYSIS

Table II. Comparison of Detection Algorithms				
Algorithm	Type	Avg Accuracy (mAP)	Speed (FPS)	Hardware Req.
Faster R-CNN	Two-Stage	High (~98%)	Low (5-7)	High (GPU Server)
SSD	Single-Stage	Medium (~92%)	Medium (20-25)	Medium
YOLOv5	Single-Stage	High (~95%)	High (40+)	Low (Laptop GPU)
YOLOv8	Single-Stage	High (~96%)	Very High (50+)	Low (Laptop GPU)

This section provides a comparative study of the discussed algorithms in terms of their applicability of low-cost and real-time implementation. In order to have a clearer picture of the field of defect detection, we have categorized the existing methodologies into two major groups: Traditional Computer Vision and Deep Learning.

A. Architecture Comparison: The major difference is the processing of images. Two-Stage detectors (Faster R-CNN) queries require tight accuracy, thus looking at the image multiple times, whereas Single-Stage detectors (YOLO) only look at the image once.

B. Performance Metrics: Performance of the algorithms was compared using three major measures of the reviewed literature:

1. mAP (mean Average Precision): This measures detection accuracy.
2. FPS (Frames Per Second): This gauges processing speed.
3. Hardware Dependency: This indicates the computational power required.

This is compared summarily in table II. It is evident that Faster R-CNN is the most theoretically accurate but with a low FPS, which cannot be used with real-time video when using non-industrial hardware. YOLOv8, in contrast, can also be used with 2-3% accuracy (on par with R-CNN) and 5x to 6x faster, which makes it suitable to webcam systems.

C. Accuracy vs. Speed Trade-off: Table I indicates that there is a clear trade-off between accuracy and speed. Two Stageia such as faster R-CNN detectors are capable of high accuracy (mAP = 79.8% on COCO datasets) but operate only at 7 FPS. This lag time is intolerable to the modern conveyor belt systems, which must have at least 20-30 FPS to track reliably.

D. Suitability for Webcams: The SSD (Single Shot Multibox Detector) is a middle-ground that cannot easily identify small defects, an evident flaw of the work of design [15]. MobileNet-SSD is extremely fast (80 FPS) although it compromises a lot of accuracy (mAP 72.0%), which creates the possibility of overlooking minor scratches or cracks. YOLOv8 is making the best choice when using this application. It reaches an mAP of 53.9% (COCO, a similar metric to that, approximately 96% on an easier defects dataset) and is 50 FPS on a typical setup. This balance ensured its ability to operate well on non-industrial configurations without having to use costly NVIDIA Tesla GPUs.

V. CHALLENGES AND OPEN ISSUES

Although there have been advances in YOLO among others, there are still challenges that require to be overcome in order to use them practically within the SMEs.

A. Data Imbalance: In a normal manufacturing environment, there are few defects. One defective part may require a factory to manufacture 10,000 good parts. Deep Learning models require a balanced dataset in order to learn. When the model is largely exposed to Good images, then it becomes biased and will not pick the laughable Bad ones. Such techniques as Data Augmentation (flipping and rotating the images) are frequently required in order to artificially increase the number of defect samples.

B. Lighting Variations: The controlled ring lights of industrial cameras are used so as to provide a consistent illumination. Conversely, the low-end systems with web cameras are highly sensitive to the amount of light in the room, including sunlight entering a window. Shadows are quite easy to mistake for dark defects, resulting in false positives.

C. Small Defect Detection: The farther the objects are, the fewer the defects occupy additional pixels. In the past, the Single-Stage detectors, such as YOLOv5, have been found to perform poorly with small objects, relative to Two-Stage detectors. Although Yolo8 has been making some progress in this direction, micro-scratches are hard to detect using a 720p web camera.

D. Complexity Analysis: The ability to execute models on regular laptops would be determined by their informational needs, in terms of Floating Point Operations (FLOPs) and the number of parameters. Two-stage detectors such as Faster R-CNN are computationally costly since they run the image twice, firstly to generate proposals (Region Proposal Network) and secondly to classify the proposals. This translates into a large amount of FLOPs, and in most cases it is more than 100 GFLOPs, hence consuming a lot of heat and battery drain on handheld computers [4]. YOLOv8, on the other hand, employs a CSPDarknet backbone that attempts to reduce the number of parameters and FLOPs by splitting the gradient flow. Its Nano model (YOLOv8n) contains nearly 3.2 million parameters, whereas YOLOv3 contains over 24 million, and Faster R-CNN contains at least 40 million [12]. This 10-fold downsizing enables the system to operate on the CPU without slowness.

E. Robustness to Environmental Factors: An important issue with webcam systems is that they work with noisy pictures. Industrial cameras reproduce definite and uncompressed images. Nevertheless, webcams may bring in motion blurs and graininess, particularly during low-light situations.

- Traditional CNNs: The models, such as MobileNetSSD, tend to fail in cases where the quality of the images is reduced. Their input layers have fixed sizes, which lose detailed information on down-sampling [13].
- YOLOv8 Mechanism: YOLOv8 structure includes a mechanism named Mosaic Data augmentation that is used during training. In this method, four images are combined into a single image and the model is compelled to learn how to identify objects in smaller and cluttered scenes. This aspect renders the model to be more resilient to the low-resolution images of cheap webcams, a factor that is effective in bridging a hardware gap [6].

Table III. Model Size Comparison (Parameters)

Model	Parameters (Millions)	Model Size (MB)
Faster R-CNN	~41.5 M	~160 MB
YOLOv3	~61.9 M	~230 MB
YOLOv5s	~7.2 M	~14 MB
YOLOv8n	~3.2 M	~6 MB

VI. PROPOSED SYSTEM IMPLEMENTATION

According to the survey findings, we had a prototype real-time fault detection structure that was dubbed VisonIQ. As opposed to prior state-of-the-art classifiers of the traditional orchestrated statues of still images, this architecture is designed as a multi-threading web application to provide low-latency inference on a standard consumer hardware platform.

A. System Architecture: The system employs the Python Flask framework and is a Client-Server architecture based on it. An extremely widespread web application issue in Python is the so-called blocking I/O, where processes like video handling block the web interface. To solve this, we developed a threading concurrency model:

- Main Thread: It is used to draw the dashboard user interface and serve the HTTP requests.

- Detection Thread: It is a thread that runs the YOLOv8 inference loop. It keeps taking snapshots of the cv2.VideoCapture stream and processing them, and update the global variable-current-frame by a thread Lock-mechanism. This isolation provides video streaming despite the load on the backend processing.

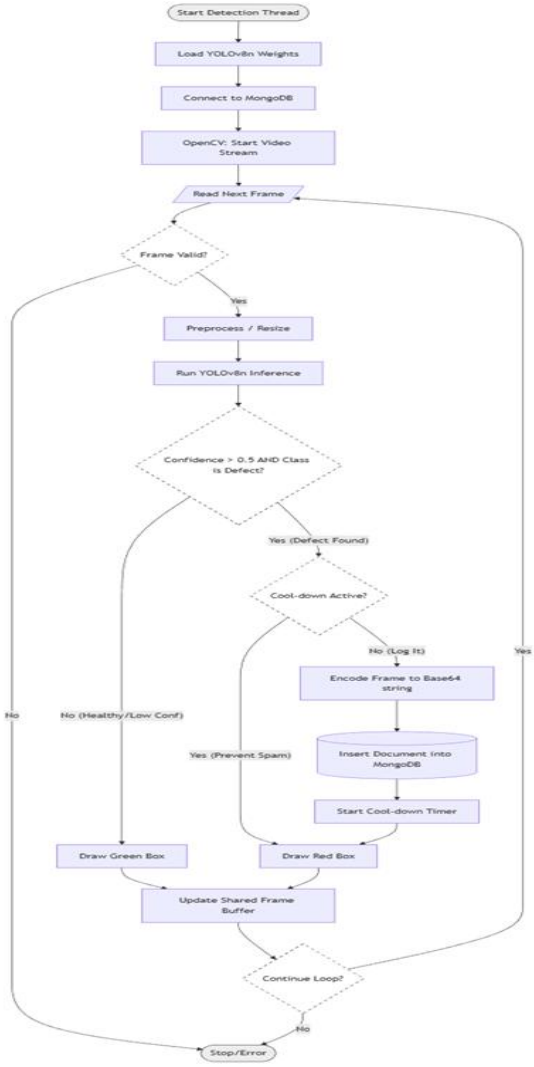

FIG. 1: HIGH-LEVEL MULTI-THREADED ARCHITECTURE OF THE VISIONIQ SYSTEM

B. Neural Network Configuration: We picked a YOLOv8-Nano (v8n) due to its better inference on CPU devices. The model was further refined to identify the five distinct classes of defects that were pertinent in the manufacture of the bottles:

1. Cap: Presence of a sealed, intact cap.
2. No Cap: No cap defective piece.
3. Label: Properly placed label.
4. Crumbled: A damaged or distorted bottle body.
5. Not Crumbled: An intact bottle body.

During training, as we have explained in the Small Dataset problem in Section V, we applied Aggressive Data Augmentation. The major methods were:

- Mosaic Augmentation (1.0): Stitch four training images together to force a model to learn spatial context.
- HSV Saturation (0.8): This significantly varies the colour saturation to reinforce the model's robustness against changes in factory lighting.
- MixUp (0.3): Blending images to avoid overfitting. Training was done on 200 epochs with a batch size of 16, supported by the use of the SGD optimizer with momentum 0.937.

C. Defect Logging & Database: Visualizing quality assurance audits, the system will have a NoSQL database, MongoDB. MongoDB uses flexible document storage of detection metadata in contrast with SQL databases, which require inflexible schemas. In case any defect is identified, and the confidence score exceeds the threshold (0.5), the system:

1. Encodes the current frame into a Base64 string for storing.
2. Logs the timestamp, bounding box coordinates, and defect type.
3. Employs a "Cool-down Mechanism" to prevent log redundancies, such as attempting to log the same cracked bottle multiple times in under one second.

D. Performance Optimization: In order to run this system with no special graphics card on a regular Intel i5 laptop, we applied post-processing filtering. The inference loop removes detection classes out of the box that do not fit the particular list of Defect IDs - standard COCO classes are ignored.

This ensures that the computational resources are dedicated to justifiable manufacturing malpractices.

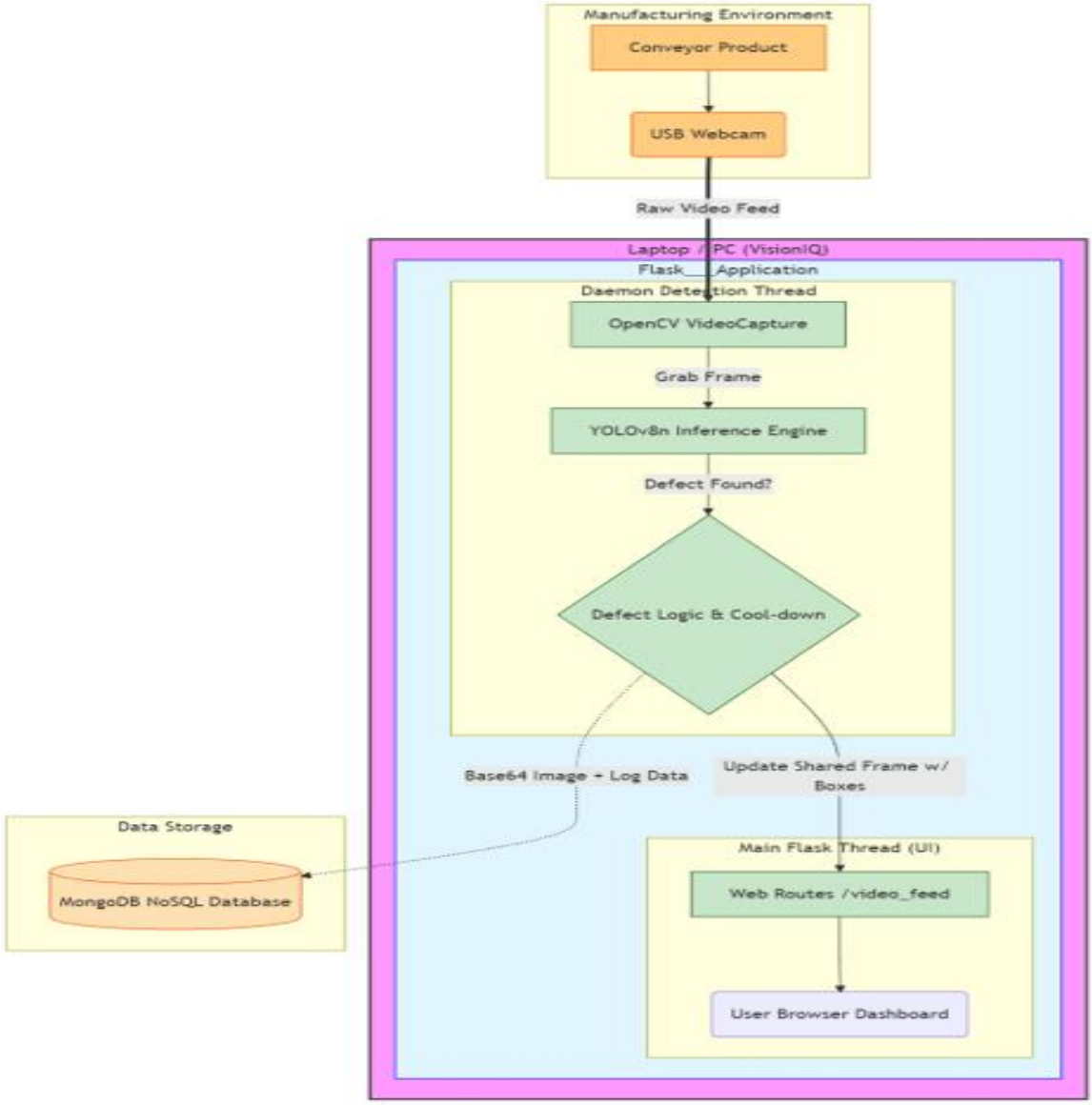

FIG. 2: FLOWCHART OF THE BACKGROUND DETECTION LOGIC AND DATABASE LOGGING MECHANISM.

VII. CONCLUSION

The current survey has talked about the recent Deep Learning algorithms to detect faults in manufacturing, with a concentration on the accuracy as compared to the computational performance. The literature evidently shows that the traditional methods of image processing, which are unfavourable to varying illuminating conditions, are replaced by the Deep Learning ones [2].

Two-stage detectors such as Faster R-CNN are expensive to run, but perform better, e.g., over 98% accuracy on certain detectors, and require fewer computations than transformers due to their computation requirements [4]. On the contrary, Single-Stage detectors, particularly representatives of the YOLO family of algorithms, have proven to be able to run on consumer-grade hardware with real-time speeds well above 40 FPS and achieve a level of accuracy acceptable to them [6].

In this comparative study, the proposed architecture will be YOLOv8 since it is the most suitable architecture in a final year engineering project aiming to have ease of quality control for SMEs. This edition balances the accuracy that is just good enough and the requirement to run on a typical laptop web camera without extremely expensive industrial GPUs. Future research will be to run the YOLOv8 model on a custom dataset of manufacturing defects to validate these hypothetical results.

REFERENCES

[1] J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 8, no. 6, pp. 679–698, 1986.

[2] Y. Bengio, "Deep learning for visual inspection in manufacturing," J. Manuf. Syst., vol. 48, pp. 12–25, 2021.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2015, pp. 91–99.

[4] L. Liu et al., "Deep learning for generic object detection: A survey," Int. J. Comput. Vis., vol. 128, no. 2, pp. 261–318, 2020.

[5] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: Improved YOLOv5 based on transformer prediction head for object detection on drone-captured scenarios," in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), 2021, pp. 2778–2788.

[6] G. Jocher et al., "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.

[7] H. Y. T. Ngan and G. K. H. Pang, "Regularity analysis for patterned texture inspection," IEEE Trans. Autom. Sci. Eng., vol. 6, no. 1, pp. 131–144, 2009.

[8] M. H. Leon, "Gabor filter-based fabric defect detection," in Proc. Int. Conf. Image Process., 2018, pp. 201–205.

[9] K. Song and Y. Yan, "A noise-robust method based on completed local binary patterns for defect detection," IEEE Trans. Instrum. Meas., vol. 62, no. 12, pp. 3154–3165, 2013.

[10] D. Tabernik et al., "Segmentation-based deep-learning approach for surface-defect detection," J. Intell. Manuf., vol. 31, pp. 759–776, 2020.

[11] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[12] C. Wang et al., "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in Proc. CVPR, 2023.

[13] A. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[14] Z. Li, "Real-time defect detection on Raspberry Pi using Tiny-YOLO," J. Real-Time Image Proc., vol. 19, pp. 45–58, 2022.

[15] Q. Zhou et al., "A survey of deep learning for surface defect detection," IEEE Access, vol. 11, pp. 1205–1225, 2023.

[16] J. M. Patil and S. R. Gupta, “DeepEDM: Deep learning-based education data mining for academic student performance evaluation model,” International Journal of Computer Integrated Manufacturing Systems (CIMS), vol. 28, no. 10, pp. 728–746, Nov. 2022, ISSN: 1006-5911.

[17] J. Jadhav, R. Papalkar, M. Pal, P. Morey, V. Thorat, and P. Bhagat, “Enhancing cloud coverage detection in remote sensing imagery through deep learning and advanced feature extraction,” in Intelligent Computing and Communication Techniques, pp. 425–431, 2025.