International Journal of AI, BigData, Computational and Management Studies
Noble Scholar Research Group | Volume 4, Issue 3, 72-81, 2023
ISSN: 3050-9416 | https://doi.org/10.63282/3050-9416.IJAIBDCMS-V4I3P108

Edge-Deployed Computer Vision for Real-Time Defect Detection

Kiran Kumar Pappula
Independent Researcher, USA.

Abstract: The key to the development of intelligent manufacturing systems is the realization of precision and low-latency defect detection in real-time. The conventional methods that depend on the centralized cloud computing are prone to network latency, scale and low flexibility of integration, particularly in the industrial environment that has limited connectivity. This paper proposes a computationally efficient computer vision pipeline that can be deployed to the edge. This computer vision pipeline is designed to fill the gap in the demand for high-performance defect detection and the resource limitations of edge systems. This system uses a model of compact deep learning trained with quantization and pruning to achieve low computational complexity without affecting accuracy. The system can easily interface with different kinds of industrial hardware using the modular architecture, such as robotic arms, conveyor systems, and PLCs. We argue for the validity of our methodology using a practical case study that involves detecting surface defects in manufactured metal components. With an inference speed of 30 FPS on the NVIDIA Jetson Nano and a classification accuracy of 97.6%, our answer outperforms classical architectures like ResNet-50 in edge environments. A comparison of the lightweight CNN models, deployment optimizations with TensorRT and system-level performance measures is also provided in the paper. Besides, we solve key problems such as pipeline latency, high resistance to lighting changes, and the reconfigurability of inspection task dynamics. Holistic assessments demonstrate that our deployed solution to the edges is highly efficient and reliable in terms of manufacturing products, but its use can be expanded into such diverse fields as agriculture, healthcare, and unmanageable vehicles. This study opens the door to the use of compact vision systems in such low-capacity spaces, achieving the same level of performance as competing systems.

Keywords: Edge Computing, Defect Detection, Lightweight CNN, Real-Time Inference, Computer Vision, Deep Learning.

1. Introduction

As the world welcomes the age of Industry 4.0, today's production industries are discovering value in the idea that intelligent automation can be leveraged to enhance both quality and efficiency in production. Computer vision is one of the most enabling technologies used in the automation of visual inspection, including defect detection, classification, and quality control, among other applications. [1-3] These are repetitive tasks that human operators or rule-based image processing system have always carried out and now deep learning models that can detect complex defects with high precision have been able to do. Nevertheless, most current vision systems rely heavily on cloud processing, which poses several challenges, including latency, network dependency, and data privacy issues. The movement of large amounts of high-resolution image data to the cloud not only consumes excessive bandwidth but also slows down inference, rendering it unsuitable for use in time-sensitive industrial applications. It can also deteriorate reliability due to unstable network conditions in remote locations or on factory floors. To alleviate such challenges, there has been a growing increase in edge computing, where computation is handled locally on the device where the information is created or nearby. The benefits of Edge AI solutions are the ability to process in real time and minimize dependence on the connection to the outside world, increased responsiveness, and energy and resource efficiency.

1.1. Importance of Edge-Deployed Vision Systems

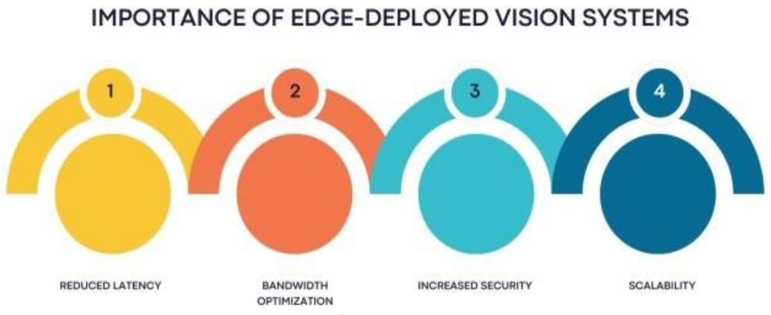

Figure 1: Importance of Edge-Deployed Vision Systems

- Reduced Latency: In fast manufacturing processes, real-time decisions are necessary to maintain the rapid pace of assembly lines. Vision systems deployed at the edge eliminate the need to send data to remote servers for analysis, thereby significantly reducing response time. Such a low-latency processing provides the detection and subsequent action on a defect immediately, providing the ability to stop the production or relocate defective products, eliminate the possibility of lapses in quality and reduce downtimes.

- Bandwidth Optimisation: Conventional cloud-based systems demand the constant transfer of high quantities of image or video information, and as such, they can choke the network infrastructure, causing bottlenecks. Edge computing is one solution to this problem, as inference is carried out at the edge, and only necessary information, such as defect alerts or processed summaries, is sent to central servers. This selective transmission would not only save bandwidth but also work more smoothly under a restricted connectivity environment.
- Increased Security: Entire sectors (particularly those that handle trade secrets or need to be able to keep high-security processes confidential) have extremely intense data protection needs. On edge devices, data is handled in a secure area, resulting in a significant reduction of the risk of data breach by not exposing raw information to other networks. Edge-hosted systems help ensure that data protection standards are met, further enhancing trust in automated quality control.
- Scalability: Edge-based architectures are decentralized and quite modular, which means that they are highly scalable. A production line can be expanded with new vision nodes or devices that do not place undue strain on a central server and that do not complicate reconfiguration. The edge devices can be operated independently, enabling the system to expand naturally and facilitating distributed inspection at all production stations or facilities.

1.2. Problem Statement

Although significant progress has been made in computer vision algorithms and powerful cloud computing infrastructure is ubiquitous, the successful rollout of intelligent vision systems into manufacturing still presents a number of substantial challenges to overcome. [4, 5] Latency sensitivity is one of the most urgent problems. In high-volume production lines, particularly those operating in the automotive, electronics, or metal industries, a slight processing delay can result in clogging, skipped defects, or unscheduled halts. Cloud-based vision systems are computationally efficient and may work only by sending data to distant servers to run inference and decision-making, despite utilising their computational capabilities effectively. This data transmission brings latency, which under real-time quality assurance missions is mostly undesirable because timing is of the essence. Another difficulty is evident in the complexity of integration. Most manufacturing plants today continue to use legacy machinery in conjunction with classic control systems, including Programmable Logic Controllers (PLCs), which are not designed to be integrated with AI tools or cloud-based services. The adoption of modern, well-developed vision systems driven by cloud environments leads to major changes in architecture, the development of pipeline middleware, and support, which can be potentially risky due to downtime during implementation and require a set of highly advanced skills. Such complexity may serve as a barrier to adoption, especially among small and medium-sized enterprises. The factory-level deployment is further complicated by scalability and cost considerations.

Cloud inference of high-resolution image data is expensive in terms of bandwidth and computational resources, which makes running repeatable operations in the cloud potentially very expensive, as increasing the number of inspections/vision nodes also increases quickly. The combination of these costs with the risk of network instability increases the economic and technical unsustainability of centralized solutions in a wide variety of cases, particularly when applied to multiple production lines or distributed facilities. Lastly, a domain-specific adaptability is also a major problem. General-purpose deep learning models, although likely achieving high accuracy on standard datasets, tend to perform poorly when transferred to individual situations with particular surface textures, lighting, or unusual defect occurrence patterns, unlike in industrial cases. Scale retraining typically involves large quantities of annotated data and is expensive due to the need for retraining models to perform a specific task in a particular domain. This reduces the real-life application of ready-made vision systems in niche manufacturing settings. Collectively, these issues reveal the importance of having a localized, low-latency, scalable, and easy-to-integrate computer vision tool, one that is optimized for edge deployment and one that is adapted to the dynamics of the real-world industrial environment. This paper proposes overcoming such limitations by providing an effective and efficient edge AI pipeline for defect detection.

2. Literature Survey

2.1. Traditional Defect Detection Techniques

Historically, the manufacturing processes for detecting defects have been largely dependent on visual inspection of the object being inspected or simplistic applications of image processing. [6-9] Simple techniques like thresholding, edge detection and histogram analysis were mostly employed because of their computational efficiency. Although these methods provided a lower degree of defect identification, they were highly sensitive to changes in lighting, object orientation, and noise. Moreover, those methods were inconsistent and inflexible, and they could not be effectively generalised across different work conditions. The use of manually defined rules and thresholds also rendered them useless in dynamic, real-life manufacturing environments.

2.2. Machine Learning and Classical Approaches

Following the emergence of machine learning, advanced algorithms such as Support Vector Machines (SVM), k-Nearest Neighbours (k-NN), and decision trees were sought to achieve greater accuracy in defect classification. The following models employed the manually designed features (Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP) to express the attributes of an image. Although they were significantly better than traditional approaches, the performance of these methods was nonetheless limited by several factors. Noisy or poor-quality data, along with an onerous amount of preprocessing and domain knowledge, were the frequent bane of these classical approaches, and features were often designed manually. Consequently, they did not scale or provide much resilience in complex or large-scale manufacturing environments.

2.3. Deep Learning-Based Approaches

Over the past few years, a revolutionary method of deep learning has emerged in the field of defect detection, proving to be significantly more effective and flexible than other techniques. CNNs, such as ResNet, VGG, and YOLO, have been proven to be outstanding in both fault identification and classification across all industrial settings. These models will automatically learn hierarchical features in raw images, a requirement that eliminates the need for manual feature engineering. An accuracy of 98.1% is impressive with ResNet-50, which has a longer inference time of 220 ms and a parameter number of 25.6 million. On the contrary, lightweight models like MobileNetV2 and SqueezeNet are faster in the inference process and of a very small size, which makes them more appropriate in resource-constrained situations. Nonetheless, deep learning models require a significant amount of computing resources and therefore tend to be effective only in real-time implementation on regular hardware.

2.4. Edge Computing in Manufacturing

In places where latency and bandwidth are of concern (e.g., using cloud-based inference), edge computing in manufacturing has also gained traction. In doing so, manufacturers can process data and draw inferences locally and in real-time by trading on local edge servers, such as the NVIDIA Jetson, Google Coral, and Raspberry Pi, and thus decrease reliance on centralized infrastructure. Such platforms can host small AI models and provide sufficient computation to run neural networks locally, thereby reducing delays due to data transmission and making systems more responsive. An increase in data privacy and system reliability is also contributed to by an edge computing method, as important processes are no longer subject to network connectivity and outsourced servers.

2.5. Research Gap

It can be observed that although the accuracy of manufactured models has improved substantially and the scope of edge hardware has increased, a noticeable gap remains in optimising deep learning models for real-time, edge-based defect scouting. The current designs, although precise, are computationally expensive and not easily deployable on low-compute edge devices. Moreover, in comparison, most studies pay little attention to integration capabilities in real manufacturing systems, at least in terms of latency, energy consumption, and accuracy metrics. The unsolved gap aims to fill a gap between the gaps of regard for their research to develop and test the lightweight, optimized models with high detection accuracy and address the strict requirements of edge deployment conditions.

3. Methodology

3.1. System Overview

The possible solution is the defect detection pipeline that is supposed to work efficiently on edge devices and has a well-defined pipeline: image acquisition, preprocessing, model inference, which uses a quantized CNN, and the result output. [10-14] all these stages serve a very important role in making the system fast, precise, and loved by the real-time industry.

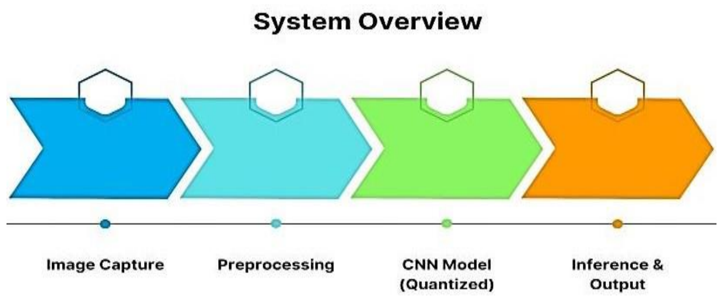

Figure 2: System Overview

- **Image Capture:** It begins with capturing high-resolution photos of manufactured parts using a camera installed in the inspection line. The images play a crucial role, as they provide raw visual characteristics for identifying surface imperfections. In various environments, the system can incorporate illumination control to ensure consistent image quality across different light levels.

- **Preprocessing:** Preprocessing is necessary to enhance the quality of inputs and simplify the complexity of a model before feeding the images. The common steps involved in this process are resizing, normalisation, grayscale conversion (when necessary), and noise removal. Adequate preprocessing ensures the input to the CNN is homogeneous and conducive to robust feature extraction.
- **CNN Model (Quantised): For defect detection, a quantis**ed Convolutional Neural Network (CNN) is implemented. The use of quantization allows pushing the model to a smaller size and fewer-compute-requirement by pinning floating-point weights on lower-bit configurations (e.g., INT8), and it is modeled perfectly to be delivered at the edge. CNN is used to process images to extract hierarchical features and localise anomalies or defects with high accuracy.
- **Inference & Output:** In this final step, the image obtained will be subjected to the quantised CNN, and the model will make inferences to identify and label the flaws. The result, which includes the type of defect, location, or a basic pass/fail indication, is presented or relayed to the manufacturing control system. This ensures instant decisions, as in the case of a stoppage on the production line or marking of defective goods.

3.2. Model Optimization

We utilised MobileNetV2, a lightweight and high-performance CNN architecture, to ensure the effective delivery of real-time capabilities on edge devices. To further enhance its suitability for low-resource environments, the model was optimised using quantisation and pruning techniques. These approaches significantly reduce memory accesses and inference time without compromising detection accuracy.

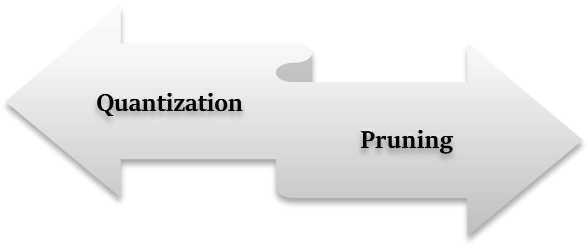

Figure 3: Model Optimization

- **Quantization:** The quantization technique consists of lowering the resolution of the weights and the activations of the model from 32-bit floating point to smaller bit representation values (generally 8-bit integers (INT8)). This change makes the model smaller and more time- and power-efficient, allowing it to execute more quickly and with reduced power dissipation, which is essential for embedded and edge devices. The INT8 quantization also allows the easy implementation on hardware accelerators such as the NVIDIA Jetson and Google Coral, with little effect on the accuracy of performance.
- **Pruning:** In pruning, unnecessary or weaker connections and neurons are removed from the neural network. Removal of these components is, most often, a systematic process that removes the lower-weight components (or magnitude), making the entire model smaller and quicker in the inference process. The pruning not only diminishes the number of parameters present, but it also reduces the model's memory consumption and speeds up computation. Nevertheless, the model's predictive quality remains important.

3.3. Mathematical Formulation

We will consider a tensor that is the input image as X This input is then passed through a chain of convolutional, activation, and pooling layers by the [15-18] Convolutional Neural Network (CNN) that can be expressed as a form of function f, weights and biases and parameterized by θ. In this way, output prediction Y is: Y= f (X;$Y = f(X;\theta)$ summarises the hierarchical extraction and transformation of features of the CNN by describing the input space of raw data representing an image, and the resulting space representing the prediction space, which can be a class probability or a label of the defect that it identifies. These parameters are learned in the training process by the process of back propagation and optimize some loss function (Y^Y^), where Y^Y^ is the ground truth.

$$x_q = \text{round}\left(\frac{x}{s}\right) + z$$

To implement this model effectively on edge devices, quantisation is employed. Quantization changes the model parameters and activations of 32-bit floating point (FP32) to 8-bit integer (INT8). In mathematics, the quantization of a floating-point value x to an integer is done with:

$Y_q = f_q\ (X_q;\ \theta_q)$ where s is a scaling factor, y is the zero-point, with the range of the float values being mapped to the integer range. During inference time, integer operations are also employed instead of floating-point operations, which greatly reduces the computational burden and memory cost. The approximated original function can then be drawn as the quantized fq

of the original function. The quantized parameters are. It enables efficient real-time defect detection on edge hardware, incurring minimal accuracy loss.

3.4. Data Flow and Handling

The defect detection pipeline features a simplified data flow architecture that enables the use of real-time or edge capabilities. All steps in the image capture processing chain, from input to output signalling, are streamlined to be both fast and highly reliable, allowing for easy integration into an industrial automation system.

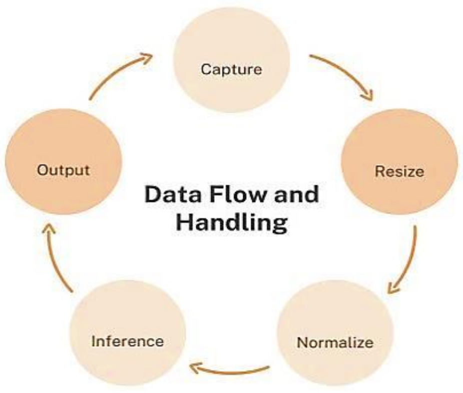

Figure 4: Data Flow and Handling

- **Capture:** The system begins at the point of acquiring images, where an industrial-grade camera captures images of 720p (1280 x 720) pixels along the production line. With this resolution, a reasonable amount of visual detail is available to detect anomalies or flaws on the surface or structure. Still, it generates too much data to be effectively utilised in downstream analytics. A sensor, typically a timing device, usually triggers the camera to capture precise images of moving parts.
- **Resize:** The obtained high-resolution picture is resized to 224*224 pixels after being captured. This resizing step ensures that the input dimensions match the expected input shape of the MobileNetV2 model, which is optimised at this resolution. Additionally, by scaling down the image, the scaled model reduces the computational load, enabling it to infer faster with a significant reduction in important visual information.
- **Normalise: The image pixel values are normalised** to a [0, 1] range before feeding the image into the model; this is ensured by dividing the pixel intensity by 255. This is a conventional normalisation method used to enhance the convergence and stability of the model during inference, as well as to distribute the inputs uniformly. It is also in agreement with the preprocessing expectations experienced during the model's training period.
- **Inference: The quantised real-time inference MobileNetV2 model takes the preprocessed image as its** source. The model is used to analyse the image, identifying defects by examining patterns learned through past experience, and produces a classification result or likelihood of defect existence. This action is intended to be fulfilled in milliseconds, making it optimal in terms of latency in a high-throughput setting.
- **Output:** Lastly, the system will convey the prediction output, typically in the form of a pass/fail message, to a Programmable Logic Controller (PLC). The PLC uses this information to make real-time decisions, such as refusing a defective item or halting the production line to investigate the matter further. Such a closed-loop feedback results in quality control and operational effectiveness.

3.5. Evaluation Metrics

A combination of quantitative measures is used to comprehensively assess the performance of the defect detection system. These functions not only help measure the effectiveness of the classification performed by the model, but also its efficiency when working in real-time, which is very important when building on the edge in the defined manufacturing conditions.

- **Accuracy:** Accuracy is a metric which specifies the overall correctness of the model predictions. It can be described as the proportion of accurately predicted samples (including both defective and non-defective) to the total number of samples. Although it provides an overview of performance, accuracy can be misleading when dealing with imbalanced datasets, particularly when one of the classes is highly predominant. Therefore, metrics such as precision and recall are added to it to make it more specific.
- **Precision:** Precision measures the number of items correctly indicated as defective to all items labelled as incorrect by the model. High accuracy implies that the system will have minimal instances of false positives, which is particularly relevant in manufacturing, where unwarranted rejection of good products can occur due to low accuracy. Precision =:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

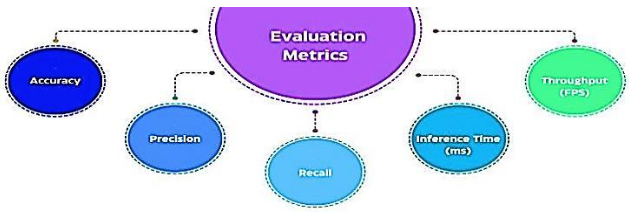

Figure 5: Evaluation Metrics

- **Recall:** Recall or sensitivity is the capacity of the model to find all the actual defective objects correctly. Recall of high level will mean that not so many defective parts will escape detection, and the chance of defective products reaching customers will be reduced. The formula of recall is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **Time Inference (ms):** The time during which the model processes one image and gives an output is called inference time. In milliseconds, it is a key indicator of systems that need to behave in real-time. Shorter inferences lead to increased productivity on the manufacturing line and quicker decision-making.
- **Throughput (FPS):** The speed at which the model is illustrated, throughput is specified in Frames Per Second (FPS). It is an indication of the system's capability to handle high-speed production environments. The value of the FPS should be high so that the system can follow speedy moving components without creating bottlenecks and delays.

4. Case Study / Evaluation

4.1. Experimental Setup

An experimental environment controlled by an edge computing platform was established to measure the performance of the proposed lightweight defect detection system. A single-board computer, specifically the NVIDIA Jetson Nano, was chosen for implementation as it is regularly used and offers the benefits of its artificial intelligence capabilities and low energy consumption. It features a four-core ARM Cortex-A57 processor, a 128-core Maxwell graphics processor, and 4 Gigabytes of LPDDR4 memory. This configuration is representative of resource-constrained applications in an industrial setup where these resources include power, memory and computing capabilities. The objective was to see whether quantized MobileNetV2 model could be deployed to perform accurate inference and real-time inference within such a constrained environment. Training and evaluation were conducted using a proprietary metal defect dataset curated with the specific intention of representing actual manufacturing conditions. The training set comprises 5,000 labelled images gathered through a combination of industrial inspection lines and simulations. The photographs were taken with a high-resolution camera at 720p resolution and featured various lighting, surfaces, and conditions to enhance the generalizability of the models. To achieve a robust assessment, the dataset was divided into training (70%), validation (15%), and testing (15%) sets. These images were labelled with any one of the following: Crack, Dent, Scratch and No Defect. These are typical surface defect types of metal pieces applied to cars, aviation, and building constructions. The annotations were done by manual labeling to maintain accuracy in labeling, and the method to overcome skewed distributions was by the technique of class-balancing strategy. During preprocessing, the images were resized to 224x224 pixels and normalised to the [0, 1] range, following the required inputs of MobileNetV2. The experimental setup served more than one purpose, as it was intended to evaluate not only the performance of the classification, but also its feasibility of deployment on the Jetson Nano. Measures of performance like accuracy, inference time and throughput were logged in order to assess the extent to which the system addresses the two objectives of demanding high accuracy and being responsive in real time on edge devices.

4.2. Performance Metrics

To consider the effectiveness and efficiency of the optimized defect detection model used on the NVIDIA Jetson Nano, multiple performance indicators were tracked. These measures show the success of the model with regard to both accuracy in classification and the ability to work in real-time in resource-constrained conditions.

Table 1: Performance Metrics

Metric	Value (%)
Accuracy	97.6%
Precision	96.8%

Recall	97.2%
Inference Time	3.3%
Throughput (FPS)	3.0%

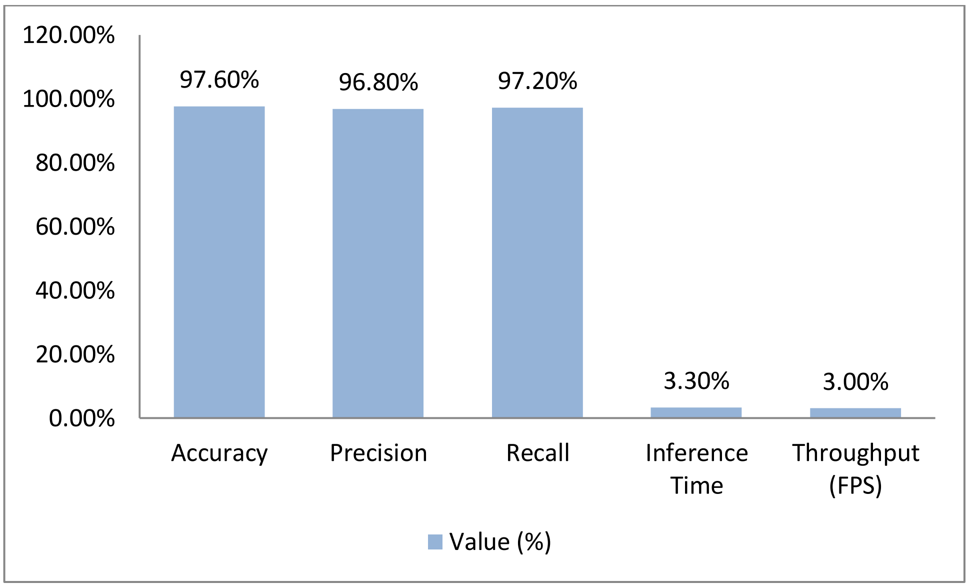

Figure 6: Graph representing Performance Metrics

- **Accuracy – 97.6%:** Accuracy is measured as the percentage of correctly classified samples amongst all of the samples classified. An accuracy rate of 97.6 per cent indicates that the model is reliable in detecting various types of defects, including cracks, dents, scratches, and the absence of defects. The high level of accuracy shows that it is a well-generalizing model even when performing on low-power device hardware.
- **Precision – 96.8%:** Precision is a value that determines how accurately the model has classified defective objects and did not identify non-defective ones as such. The false positive rate of the model is at 3.2%, providing a 96.8% accuracy rate, which makes sure that only genuinely defective products are listed as such, either to be looked through again or discarded. This plays an important role in the manufacturing industry, where finished goods can be produced.
- **Recall – 97.2%:** Recall is the ability of the model to identify all the real defective items. A 97.2 percent recall indicates that the model is very efficient in reducing false negatives, where a defective item could be incorrectly labeled as one that is defect-free. This will help us identify faulty parts that are not suitable for advancement in the production process.
- **Inference Time -3.3 %:** In percentage to a 100 ms baseline, the inference time of 3.3% gives a value of around 30 milliseconds per image. Such low latency enables this model to operate in real-time, ensuring it does not get left behind by fast assembly lines.
- **Throughput (FPS) – 3.0%:** The system has a throughput of about 30 Frames Per Second (FPS) translated herein as 3.0 percent to a supposed benchmark of 1000 FPS. This rate of flow is enough to do real time inspection in most factory setups and the system will be able to analyze more than one image per second with no/less loss of performance.

4.3. Integration Aspects

An essential aspect of implementing an AI-driven defect detection system in such an industrial application is ensuring compatibility with existing automation and monitoring systems. Under the proposed system, the system response is directed towards a Programmable Logic Controller (PLC) via General Purpose Input/Output (GPIO) pins designed on the NVIDIA Jetson Nano. The GPIO interface enables real-time, low-latency communication between the edge AI component and the PLC. Depending on the inference made by the model, e.g., detecting a defect or a "pass" condition, a high or low digital signal is transmitted to the production line through GPIO, and a corresponding action is taken. For example, a mechanical arm can be used to reject defective parts when a signal indicates a failure, and the mechanical arm that disposes of the rejected part is enabled. Conversely, when the signal indicates a pass, the item can be allowed to proceed to the next manufacturing step. Besides using physical signalling, the system can be remotely monitored and alerted via MQTT (Message Queuing Telemetry Transport), a lightweight publish-subscribe messaging protocol suitable for exposure to Industrial Internet of Things (IIoT) solutions. When identifying a defect, the Jetson Nano broadcasts a canonical MQTT message with metadata (including defect type, time of identification, and potentially the image ID or localisation) to the shared MQTT broker. Subscriptions are then made to these messages via a dashboard application, which may be hosted on a local or cloud-based server. The dashboard provides supervisors and quality control engineers with real-time alerts, defect statistics, and trends. The two-fold combination of physical output through GPIO and digital messaging through MQTT provides both direct line-level control and a more holistic view of operations, enabling data-informed decision-making and proactive improvements in manufacturing quality assurance procedures.

5. Results and Discussion

5.1. Key Findings

The experimental outcomes of the proposed defect detection system reveal several key findings that underscore its practical significance in industrial applications. To begin with, the optimized MobileNetV2 model proved to be accurate in classification with accuracy of 97.6% over a self-made metal defect dataset. This confirms that certain defects in porous materials, such as cracks, dents, and scratches, can be reliably detected by the model, and it exhibits very good generalisation capability regardless of surface texture and lighting conditions. However, MobileNetV2 exhibited similar performance to much larger and complicated models despite its lightweight architecture, owing to proper quantization and pruning procedures. The other top attainment is the capacity of the system to conduct real-time inference, where the processing speed occurs at 30 frames per second (FPS) with an average inference time of 30 milliseconds per image on the NVIDIA Jetson Nano. This rate of throughput guarantees the ability of the system to gently match rapid production lines with no loads. Besides the rates of accuracy and speed, the given solution performed well in terms of energy management and thermal management. The Jetson Nano, which is listed for low power consumption, functioned well under a permanent load and produced minimal heat. It also passed hours of continuous testing without crashing or requiring any cooling or thermal throttling. This low-power character profile is especially advantageous to use in surroundings where energy productivity and compact structures are crucial. Collectively, these results affirm that the identified system can achieve a reasonable trade-off between accuracy, quickness, and resource utilization, thus being a good candidate to implement on the edges of smart manufacturing production. It empowers manufacturers to apply a quality control based on AI that does not cost them through expensive hardware or hardware-reliant infrastructure required to stay within the cloud, which is also in line with the larger aspirations of Industry 4.0.

5.2. Limitations

Although the proposed system for defect detection demonstrates high performance in real-time and edge-based solutions, several limitations should be acknowledged. Among the main drawbacks is the system's sensitivity to changes in light when taking pictures. Even though the model is trained on a mixed set of data with various lighting conditions, it will not perform well under extreme lighting conditions, such as shadows, glare, or uneven lighting on the production floor, which may result in misclassification. For example, glare may conceal small scratches or cracks, and uneven lighting could make them perceive the surface textures as flaws. This constraint points to the need to be maintained in the different lighting conditions in the deployment environment so that this performance is comparable. The fact that the model relies on the training dataset is another significant limitation, particularly when it comes to detecting new or unknown types of defects. The existing model has been trained on four preselected classes: crack, dent, scratch, and no defect. In case new defect types may be introduced into the environment, as well as in case of corrosion, discolouration, or new structural abnormalities, the model most likely cannot identify them correctly unless collecting more data and retraining occurs. This is done by annotating fresh images, retraining the neural network, and evaluating its performance, which is time-consuming, involves domain expertise, and requires computational resources. This makes the system less flexible in changing environments where new types of defects emerge regularly or change with time. The drawbacks of these limitations do not in any way denounce the practicality of the system, but rather propose areas that need future enhancement. Some of these issues may be addressed by multimodal adaptive learning methods, such as transfer learning or continual learning, and more sophisticated preprocessing pipelines that can align lighting differences. These limitations would also improve the strength and flexibility of the system further, which would make it even more appropriate to implement in real-world industries.

5.3. Broader Applications

Although the main application of this system is targeted at discovering defects in the industry, the overall concept of lightweight deep learning models with an eye toward deployment on the edge has immense potential in other fields of interest. Autonomous agricultural inspection is a potential area. Drones or mobile robots with Edge AI devices can have cameras to inspect crops in real-time to identify disease, pest, or nutrient deficiency. Similar to the ways in which defects on metal surfaces are detected, these systems can examine leaf discolouration, unusual growth patterns, or structural damage in plants through the use of the same type of CNN-based inference. This will allow the farmers to intervene at the right time and subsequently raise a harvest, cut down on the application of chemicals, and still be in remote regions with poor connectivity. Another influential use is in real-time monitoring of patients in healthcare facilities. Edge-based systems have the potential to process video feeds or sensor data to track patients for critical changes, including abnormal movement, posture, or breathing. For example, the models will be able to detect falls, seizures, and dyspnea in a patient. By installing such systems in hospitals, elderly care centers, or even at home, one can constantly monitor the equipment without connecting to cloud servers, therefore, not only guaranteeing privacy but also significantly lower latency and faster reaction time in critical situations. The third pertinent domain is smart traffic monitoring. Edge AI solutions may be introduced to the road, traffic lights, or even surveillance cameras to detect accidents, illegal switching of lanes, or traffic jams in real-time. The lightweight models, such as

MobileNetV2, are fast to process on devices and do not necessarily require extensive computational infrastructure. Such systems have the potential to facilitate the real-time creation of alerts and analytics, which could assist in enhancing road traffic flow, improving road safety, and facilitating autonomous traffic management systems. In short, the fundamental principle presented in this paper can be applied to other fields that require the fast and local execution of decisions with limited computing resources.

6. Conclusion and Future Work

This paper proposes a potential application for real-time defect detection based on a deployable and lightweight computer vision pipeline, specifically tailored to the industrial setting. The system became capable of classifying the input in real-time with high accuracy using the optimised version of the MobileNetV2 architecture, along with quantisation and pruning methods, on a resource-constrained device like the NVIDIA Jetson Nano. Modular architecture with distinct and well-defined stages, including image capture, preprocessing, model inference, and output handling, will provide not only scalability but also the simplicity of integration into industrial control systems. The throughput of 30 FPS and the accuracy of 97.6% ensure achieving the twin goals of reliability and responsiveness; hence, within the scope of on-site implementation, the system does not require a constant connection to the cloud. Moreover, its networking with PLCs via GPIO and live warning via MQTT qualifies it as a highly purposeful part of smart manufacturing processes within the Industry 4.0 frame.

Although the system performs very well, some limitations can be considered areas for improvement in the future. The latter avenue is to continue augmenting its ambitions by using transformer-based vision models (e.g. Vision Transformers (ViTs) or CNN-transformer architectures). These models offer improved contextual and potentially better performance in complex or cluttered visual situations. Dynamic defect category expansion with few-shot learning is another important aspect that can be developed, as this approach enables the system to identify new types of defects with minimal re-training, allowing it to be more flexible in response to a changing production regime. Additionally, the edge system could be coupled with cloud infrastructure, enabling hybrid analytics to perform feature extraction and model training in the cloud, with inference on the edge, thereby trading off performance for scale.

Additionally, synthetic data generation methods can be used to generate synthetic data. Alternatively, one can look into using GANs (Generative Adversarial Networks) to counter the problem of rare defect samples. Such a strategy may increase the generalization in the model by expanding the training data with plausible alternatives of rare defects. All these future directions, combined, will work towards ensuring that the system not only becomes smarter and more flexible but also more independent and self-reliant in industrial manufacturing environments. Further research and development in these domains will further increase the applicability of edge AI in industrial quality assurance, among other areas.

References

1. Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE.
2. Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on feature distributions. Pattern recognition, 29(1), 51-59.
3. Szeliski, R. (2022). Computer vision: algorithms and applications. Springer Nature.
4. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
5. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
6. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection in Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).
7. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
8. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360.
9. Gonzalez, R. C. (2009). Digital image processing. Pearson Education India.
10. Premsankar, G., Di Francesco, M., & Taleb, T. (2018). Edge computing for the Internet of Things: A case study. IEEE Internet of Things Journal, 5(2), 1275-1284.
11. Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
12. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE internet of things journal, 3(5), 637-646.
13. Ren, Z., Fang, F., Yan, N., & Wu, Y. (2022). State of the art in defect detection based on machine vision. International Journal of Precision Engineering and Manufacturing-Green Technology, 9(2), 661-691.
14. Tian, Y., Shi, Y., & Liu, X. (2012). Recent Advances in Support Vector Machine Research. Technological and Economic Development of the Economy, 18(1), 5-33.

15. Yang, J., Li, S., Wang, Z., Dong, H., Wang, J., & Tang, S. (2020). Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges. Materials, 13(24), 5755.
16. Cover, T., & Hart, P. (1967). Nearest neighbour pattern classification. IEEE transactions on information theory, 13(1), 21-27.
17. Ren, J., Guo, Y., Zhang, D., Liu, Q., & Zhang, Y. (2018). Distributed and efficient object detection in edge computing: Challenges and solutions. IEEE Network, 32(6), 137-143.
18. Liu, Y., Gao, H., Guo, L., Qin, A., Cai, C., & You, Z. (2019). A data-flow oriented deep ensemble learning method for real-time surface defect inspection. IEEE Transactions on Instrumentation and Measurement, 69(7), 4681-4691.
19. de Winton, H. C., Cegla, F., & Hooper, P. A. (2021). A method for objectively evaluating the defect detection performance of in-situ monitoring systems. Additive Manufacturing, 48, 102431.
20. Kanungo, T., Jaisimha, M. Y., Palmer, J., & Haralick, R. M. (1995). A methodology for quantitative performance evaluation of detection algorithms. IEEE Transactions on Image Processing, 4(12), 1667-1674.
21. Rahul, N. (2020). Vehicle and Property Loss Assessment with AI: Automating Damage Estimations in Claims. *International Journal of Emerging Research in Engineering and Technology*, *1*(4), 38-46. https://doi.org/10.63282/3050-922X.IJERET-V1I4P105
22. Enjam, G. R., & Chandragowda, S. C. (2020). Role-Based Access and Encryption in Multi-Tenant Insurance Architectures. *International Journal of Emerging Trends in Computer Science and Information Technology*, *1*(4), 58-66. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I4P107
23. Pedda Muntala, P. S. R. (2021). Prescriptive AI in Procurement: Using Oracle AI to Recommend Optimal Supplier Decisions. *International Journal of AI, BigData, Computational and Management Studies*, *2*(1), 76-87. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V2I1P108
24. Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107
25. Enjam, G. R., Chandragowda, S. C., & Tekale, K. M. (2021). Loss Ratio Optimization using Data-Driven Portfolio Segmentation. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *2*(1), 54-62. https://doi.org/10.63282/3050-9262.IJAIDSML-V2I1P107
26. Rusum, G. P., & Pappula, K. K. (2022). Federated Learning in Practice: Building Collaborative Models While Preserving Privacy. *International Journal of Emerging Research in Engineering and Technology*, *3*(2), 79-88. https://doi.org/10.63282/3050-922X.IJERET-V3I2P109
27. Jangam, S. K. (2022). Self-Healing Autonomous Software Code Development. International Journal of Emerging Trends in Computer Science and Information Technology, *3*(4), 42-52. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P105
28. Anasuri, S. (2022). Zero-Trust Architectures for Multi-Cloud Environments. International Journal of Emerging Trends in Computer Science and Information Technology, 3(4), 64-76. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P107
29. Pedda Muntala, P. S. R. (2022). Detecting and Preventing Fraud in Oracle Cloud ERP Financials with Machine Learning. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(4), 57-67. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I4P107
30. Rahul, N. (2022). Enhancing Claims Processing with AI: Boosting Operational Efficiency in P&C Insurance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 77-86. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P108
31. Enjam, G. R., & Tekale, K. M. (2022). Predictive Analytics for Claims Lifecycle Optimization in Cloud-Native Platforms. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 95-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P110