THE UNIVERSITY OF DANANG
UNIVERSITY OF SCIENCE AND TECHNOLOGY
FACULTY OF PROJECT MANAGEMENT

CAPSTONE PROJECT

Major: Industrial Management

Title:

AN APPLICATION OF CONVOLUTIONAL NEURAL NETWORK FOR AUTOMATED DEFECT DETECTION IN BLUETOOTH SPEAKER MANUFACTURING

Supervisor : **Ph.D NGUYEN THI PHUONG QUYEN**
Student : **VO THI NGOC Y**
Class : **20QLCN1**
Student ID : **118200183**

Da Nang, June 24, 2025.

THE UNIVERSITY OF DANANG

UNIVERSITY OF SCIENCE AND TECHNOLOGY

FACULTY OF PROJECT MANAGEMENT

CAPSTONE PROJECT

Major: Industrial Management

Title:

AN APPLICATION OF CONVOLUTIONAL NEURAL NETWORK FOR AUTOMATED DEFECT DETECTION IN BLUETOOTH SPEAKER MANUFACTURING

Supervisor : **Ph.D NGUYEN THI PHUONG QUYEN**

Student : **VO THI NGOC Y**

Class : **20QLCN1**

Student ID : **118200183**

Da Nang, June 24, 2025.

PROJECT SUMMARY

With increasing demands for quality and consistency in Bluetooth speaker manufacturing, this project applies deep learning techniques, specifically Convolutional Neural Networks (CNNs), to automate the visual defect inspection process. The system replaces manual inspection by using industrial cameras and a motorized turntable to capture images from different angles. This allows for accurate and consistent defect detection.

The main objective is to reduce inspection time, minimize human error, and improve production efficiency. The model is trained to identify surface defects such as scratches, dirt, and misalignments, focusing on five critical areas of the speaker: the handle, IO cover, logo area, PR surfaces, and silicone buttons.

Although the system performs well in detecting external defects, its effectiveness depends on the quality and diversity of training data. It does not evaluate internal components or acoustic performance.

This thesis is structured into six chapters:

- Chapter 1: Overview of the Capstone Project
- Chapter 2: Theoretical Basis
- Chapter 3: Current Situation Analysis
- Chapter 4: Convolutional Neural Network for Automated Defect Detection
- Chapter 5: Model Performance Analysis
- Chapter 6: Conclusion and Recommendations

CAPSTONE PROJECT ASSIGNMENT

Student's Full Name: Vo Thi Ngoc Y

Student ID: 118200183 Class: 20QLCN1

Faculty: Faculty of Project Management Major: Industrial Management

1. Project title: An Application of Convolutional Neural Network for Automated Defect Detection in Bluetooth Speaker Manufacturing

2. The project is subject to an intellectual property agreement regarding its outcomes. ☐

3. Initial Data and Figures: A labeled image dataset from Bluetooth speaker inspection, including six defect types.

4. Main content and analysis sections

Chapter 1: Overview Of the Capstone Project

Chapter 2: Theoretical Nasis

Chapter 3: Current Situation Analysis

Chapter4: Convolutional Neural Network for Automated Defect Detection

Chapter 5: Model Performance Analysis

Chapter 6: Conclusion And Recommendations

5. Name of the supervisor: Nguyen Thi Phuong Quyen

6. Date of project assignment: 17/02/2025

7. Date of project completion: 16/06/2025

Da Nang, June 24, 2025

Head of Department **Project Supervisor**

Ph.D Huynh Nhat To **Ph.D Nguyen Thi Phuong Quyen**

PREFACE

This capstone project marks the final stage of my undergraduate journey at the Faculty of Project Management, Danang University of Science and Technology. Throughout the research and implementation process, I had the opportunity to apply the knowledge and skills acquired during my academic studies into a practical and meaningful project.

The project focuses on applying Convolutional Neural Networks (CNN) to automate defect detection in Bluetooth speaker manufacturing. This topic not only reflects the growing trend of integrating artificial intelligence into industrial production but also addresses practical challenges in improving inspection efficiency and product quality.

I would like to express my sincere gratitude to my academic supervisor, Ms. Nguyen Thi Phuong Quyen, for her valuable guidance and continuous support throughout this project. I also extend my thanks to GGEC (VIETNAM) CO., LTD, where I had the opportunity to observe and collect practical data that greatly contributed to the success of this study.

Despite my best efforts, due to limited time and experience, this report may still contain certain shortcomings. I respectfully welcome any feedback and suggestions for improvement.

DaNang, June 14, 2025

Student

Vo Thi Ngoc Y

DECLARATION

My name is Vo Thi Ngoc Y, a student of class 20QLCN1. I hereby declare that:

- This graduation thesis is the result of my own study and research, based on theoretical knowledge and actual data collected from the company, carried out under the guidance of my academic advisor.
- The thesis is entirely original and is the outcome of my individual effort.
- All references and citations used in this thesis are properly acknowledged and listed in the References section.
- I take full responsibility for any violations related to plagiarism or non-compliance with the university's regulations.

Da Nang, June 14, 2025

Student

Vo Thi Ngoc Y

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CNN	Convolutional Neural Network
IO	Input/Output
mAP	mean Average Precision
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
DPPM	Defective Parts Per Million
API	Application Programming Interface
FPR	False Positive Rate
TPR	True Positive Rate
ROC Curve	Receiver Operating Characteristic Curve
AUC	Area Under the Curve
ReLC	Rectified Linear Unit

CHPATER 1: OVERVIEW OF THE CAPSTONE PROJECT

1.1 Problem statement

In the manufacturing process of consumer electronic devices such as Bluetooth speakers, maintaining consistent product appearance is a critical factor that directly affects user experience and brand perception. Currently, visual inspection in Bluetooth speaker production is mostly performed manually by workers. However, this method has several limitations, including human error, fatigue, subjectivity, and a lack of consistency—especially when production volumes increase.

As quality standards continue to rise and production speeds accelerate, relying entirely on manual inspection is becoming less practical. Defects such as scratches, misalignment, minor assembly flaws, or logo displacement can be easily overlooked by the human eye. This not only impacts product quality but also increases production costs due to rework or rejection of defective units.

In response to these challenges, automated visual inspection (AVI) systems, combined with deep learning technologies—particularly Convolutional Neural Networks (CNNs)—have emerged as promising alternatives to manual inspection. CNNs have demonstrated outstanding performance in image recognition and object detection tasks. However, applying these models in real manufacturing environments, especially for products with complex surfaces like Bluetooth speakers, still presents a number of difficulties.

This research focuses on developing a CNN-based solution for detecting surface defects in Bluetooth speaker production. The goal is to explore a more reliable and efficient approach to visual inspection—one that reduces human dependency, enhances accuracy, and supports scalability in modern manufacturing settings.

1.2 Project title

With the growing demand for quality and consistency in electronic device manufacturing, especially in Bluetooth speakers, this project focuses on applying deep learning techniques to improve the visual inspection process. Therefore, the title of the project is:

An Application of Convolutional Neural Network for Automated Visual Defect Detection in Bluetooth Speaker Manufacturing.

1.3 Objectives

This project aims to apply deep learning to build an automated visual inspection system for Bluetooth speakers. The system replaces traditional manual visual inspection

with an automated inspection process using industrial cameras and a motorized turntable. By capturing consistent images from multiple angles and analyzing them with deep learning models, the system accurately detects surface defects, significantly reducing inspection time, minimizing human error, and improving overall production efficiency.

1.4 Scope and limitations

This project focuses on developing a deep learning-based visual inspection model tailored for Bluetooth speakers. The model is trained to detect surface defects such as scratches, dirt, misalignments, and deformations on five critical areas of the product: the handle, IO cover, logo, PR surfaces, and silicone buttons.

The inspection system utilizes industrial cameras and a motorized turntable to automatically capture images from different angles, allowing the model to analyze and evaluate multiple sides of the speaker efficiently.

However, the model's effectiveness depends heavily on the quality, quantity, and variability of training data. It may have limited ability to detect new or uncommon defects not represented in the dataset. Additionally, the system is designed solely for external surface inspection and does not cover internal electronics or acoustic performance.

1.5 Project Structure

This Capstone Project is structured into six chapters, each addressing a key aspect of the project titled “An Application of Convolutional Neural Network for Automated Defect Detection in Bluetooth Speaker Manufacturing”.

Chapter 1: Overview Of the Capstone Project

Chapter 2: Theoretical Nasis

Chapter 3: Current Situation Analysis

Chapter4: Convolutional Neural Network for Automated Defect Detection

Chapter 5: Model Performance Analysis

Chapter 6: Conclusion And Recommendations

CHAPTER 2: THEORETICAL BASIS

2.1 Overview of Visual Inspection in Industrial Products

Visual inspection plays a crucial role in the quality control process of industrial products. Its primary task is to detect surface defects such as cracks, scratches, deformations, dents, contamination, and other morphological abnormalities, ensuring that the product meets technical and aesthetic standards before reaching the market. [1]

In traditional manufacturing systems, visual inspection is predominantly performed manually by human operators. While this method offers flexibility, it is often influenced by subjective factors such as operator fatigue, personal experience, and working environment conditions, leading to inconsistency and reduced reliability in inspection results. [2]

The advancement of computer vision and artificial intelligence technologies has driven the adoption of automated visual inspection systems in industrial production. These systems utilize image sensors, lighting devices, and image processing algorithms to automatically detect, classify, and evaluate surface defects on products with high speed and accuracy.

The application of automated visual inspection brings numerous benefits, including enhanced accuracy, reduced dependency on human labor, improved productivity, and consistent quality control. However, automated inspection systems also face several challenges, such as the need for high flexibility to accommodate diverse product surfaces and the complexity of identifying various types of defects.

In this context, deep learning techniques, particularly Convolutional Neural Networks (CNNs), have been extensively researched and applied to improve the efficiency and reliability of automated visual inspection systems.

2.2 Applications of Convolutional Neural Networks in Industrial Visual Inspection

In the field of industrial visual inspection, Convolutional Neural Networks (CNNs) have been widely adopted due to their ability to automatically extract image features and perform classification with high accuracy. Recent studies have demonstrated the superior performance of CNNs in detecting surface defects, assembly issues, and microstructural anomalies that traditional methods often miss.

A notable example is the work of Jiang et al. (2025), who proposed the Defect R-CNN model, an enhanced version of Mask R-CNN, designed to improve defect detection accuracy in industrial CT images. The model integrates an Edge-Prior Convolutional Block (EPCB) that enables the network to focus on defect boundaries—

areas typically difficult to identify. Experimental results showed a mean Average Precision (mAP) of 0,983 for bounding boxes and 0,956 for segmentation masks, with an impressive processing speed of 76,2 FPS, making it suitable for real-world industrial applications. [3]

Wang and Yu (2022) developed a hybrid architecture called SSD-Faster Net, combining the fast region proposal capability of SSD with the high accuracy of an improved Faster R-CNN. The two-stage network first detects candidate regions using SSD, followed by detailed analysis with Faster R-CNN. The model achieved an mAP of 84,03%, a 13,42% improvement over the baseline Faster R-CNN, and a 7% increase in inference speed, proving its applicability in high-speed production lines. [4]

In the context of printed circuit board (PCB) manufacturing, Calabrese et al. (2025) compared Mask R-CNN and YOLOv8 for defect detection tasks such as short circuits and missing drill holes. Using two independent datasets, they found that Mask R-CNN excels in segmentation accuracy, while YOLOv8 offers superior real-time performance, highlighting that model selection should align with specific application requirements. [5]

Another important contribution is from Kim and Kim (2020), who developed a lightweight CNN model using an encoder–decoder architecture with bottleneck layers, reducing the parameter count for deployment on resource-constrained embedded devices. The model achieved 99,9% accuracy on the DAGM dataset (MobileNetV2@0.35) and 99,33% on the NEU Surface Defect dataset (Ours@0.35). [6]

In the automotive industry, García-Pérez et al. (2022) presented an Automatic Defect Recognition (ADR) system using CNNs for analyzing industrial X-ray images. Trained on the GDXray dataset, the model achieved an mAP of 94,2% at IoU=50%, with inference times under 400 ms per DICOM image, making it feasible for real-time production environments. [7]

Similarly, Wei et al. (2021) proposed a deep CNN-based method for automatic steel surface defect detection and classification. The architecture, based on Faster R-CNN, integrates techniques such as Deformable Convolutional Networks (DCN), Feature Pyramid Networks (FPN), and CoordConv layers to enhance performance. The model achieved mAP of 87,6% and average inference time of 2,9 ms per image, outperforming traditional approaches. [8]

For MEMS pressure sensors, Deng et al. (2022) introduced a novel defect inspection system based on CNNs. Built upon Faster R-CNN and enhanced with

random data augmentation and a custom defect classifier, the system achieved an mAP of 92,39% and classification accuracy of 97,2%, meeting the precision requirements of modern production lines. [9]

Through the review of related studies, it is evident that Convolutional Neural Networks (CNNs) have been widely and effectively applied in industrial visual inspection, particularly in fields such as steel manufacturing, electronic components, MEMS sensors, and X-ray imaging. CNN-based models not only offer high accuracy but also meet practical requirements in terms of inference speed, scalability, and real-world deployment.

However, there is currently a lack of research focusing on the application of CNNs for the visual inspection of Bluetooth speakers - a popular consumer product that requires strict appearance quality control. This presents a promising opportunity for developing an automated inspection system using CNNs tailored to the specific characteristics and surface defects of this product line.

2.3 Model Performance Evaluation

2.3.1 Confusion Matrix

In the field of machine learning, especially in classification tasks, it is essential to quantitatively evaluate the performance of a model. One of the most effective and widely used tools for this purpose is the confusion matrix. This matrix provides a clear visualization of the model's performance by comparing the predicted labels with the actual ground truth values across different classes.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

Figure 1. Confusion Matrix Illustration [10]

The confusion matrix is typically structured in a 2x2 format for binary classification problems, as illustrated in the figure below. It consists of four key components:

- **True Positive (TP)**: The number of instances that are correctly predicted as positive and are actually positive.
- **False Positive (FP)**: The number of instances that are incorrectly predicted as positive, while in reality they are negative.
- **True Negative (TN)**: The number of instances that are correctly predicted as negative and are actually negative.
- **False Negative (FN)**: The number of instances that are incorrectly predicted as negative, while in reality, they are positive.

These four components are essential for calculating performance metrics such as accuracy, precision, recall, and F1-score, which are further used to assess the overall effectiveness of the classification system. Each metric provides specific insights. For example, a high number of false negatives in this context could lead to defective Bluetooth speakers reaching consumers, which may result in customer dissatisfaction and increased warranty costs.

In this research, the confusion matrix plays a key role in evaluating the effectiveness of the visual defect classification model integrated into the automated inspection system on the production line of Bluetooth speakers. By interpreting the matrix values, areas where the model performs well or requires improvement can be clearly identified, supporting the refinement of the quality assurance process.

2.3.2 Performance Metrics

In the scope of this study, four primary evaluation metrics are utilized: accuracy, precision, recall, and F1-score. These metrics are complementary and reflect different aspects of the performance of the classification model. The following sections provide a detailed explanation of each metric, including its meaning, calculation method, and application in the context of automated visual inspection.

2.3.2.1 Accuracy

In the process of evaluating the effectiveness of a classification model, particularly during testing and model comparison, accuracy is often the first metric to be considered. It is a simple yet effective indicator that provides an overall view of how well the model performs across the entire dataset.

Accuracy is defined as the ratio between the number of correctly predicted instances and the total number of predictions. Specifically, it is calculated using the following formula:

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \text{ [10]}$$

Where TP (True Positive) and TN (True Negative) represent the correctly predicted instances of the positive and negative classes, respectively, and FP (False Positive) and FN (False Negative) represent the misclassified cases. Therefore, the accuracy metric reflects the model's overall ability to correctly classify both categories.

In terms of advantages, accuracy is highly intuitive and easy to compute, especially useful when the dataset has a relatively balanced class distribution. It is commonly employed as an initial benchmark for general model evaluation and as a basis for selecting candidate models during early stages.

However, despite these benefits, accuracy also exhibits some clear limitations. One of the most significant drawbacks is its lack of sensitivity when applied to imbalanced datasets, where one class heavily outweighs the other. In such cases, a model may achieve high accuracy while performing poorly in identifying the minority class, which may carry higher importance in practical scenarios such as product defect detection.

Therefore, in applications that require high reliability and sensitivity—such as automated visual inspection in manufacturing—accuracy should be regarded as a reference metric only. It should be complemented with other performance indicators such as precision, recall, and F1-score to obtain a more complete and accurate evaluation of the model's classification performance.

2.3.2.2 Precision

Precision plays a key role in evaluating the accuracy of the model's positive predictions. Precision measures the proportion of instances that were predicted as positive (e.g., defective) and are actually positive, thus indicating the model's ability to minimize false alarms (false positives). The precision metric is calculated using the following formula:

$$\textbf{Precision} = \frac{TP}{TP+FP} \text{ [10]}$$

Where:

- TP (True Positive): the number of defective samples correctly identified.

- FP (False Positive): the number of non-defective samples incorrectly predicted as defective.

A high precision value implies that when the system raises an alert, it is very likely that the detected sample is truly defective. This is especially important in production environments where mistakenly rejecting good products not only reduces operational efficiency but also leads to increased material waste and unnecessary labor costs for re-inspection.

However, like other metrics, precision also has limitations. A model can achieve high precision by reducing the number of positive predictions, meaning it only labels a small number of samples as defective to ensure they are truly faulty. While this boosts precision, it may severely impact recall, leading to a higher number of undetected defects. For this reason, precision should always be interpreted in conjunction with other metrics—particularly recall—to obtain a balanced understanding of the model's behavior.

2.3.2.3 Recall

Recall, also referred to as the True Positive Rate or Sensitivity, is a crucial metric in evaluating the performance of classification models, particularly in scenarios where missing a positive instance may lead to serious consequences. Recall measures the model's ability to correctly identify all relevant positive cases, i.e., instances that truly belong to the target class. The formula for calculating recall is as follows:

$$\textbf{Recall} = \frac{TP}{TP+FN} \text{ [10]}$$

Where:

- TP (True Positive): the number of actual positive samples correctly classified.
- FN (False Negative): the number of actual positive samples incorrectly classified as negative.

Recall indicates how effectively the model avoids overlooking important positive instances. This metric is especially valuable in applications where failing to detect a defect or anomaly can lead to safety risks, financial losses, or reduced product quality—such as medical diagnosis, fraud detection, or visual inspection in industrial manufacturing.

A model with high recall demonstrates strong capability in detecting positive cases, ensuring that the majority of defective products are identified and appropriately handled. However, high recall does not guarantee that all positive predictions are correct.

In efforts to maximize recall, the model might classify more samples as positive including many false positives, thereby reducing precision.

For this reason, recall should not be evaluated in isolation. Balancing precision and recall are essential to ensure that the model not only identifies positive cases reliably but also avoids excessive false alarms. This balance is captured by the F1-score, which will be discussed in the following section.

2.3.2.4 F1-Score

F1-score is a metric used to evaluate the performance of classification models. It combines two important measures: precision and recall. In many cases, focusing on only one of these measures can cause an imbalance in the model's performance. The F1-score helps to give a more balanced view by calculating the harmonic means of precision and recall. The formula for F1-score is:

$$\textbf{F1-score} = 2* \frac{Precision * Recall}{Precision+Recall} \text{ [10]}$$

F1-score ranges from 0 to 1. A higher score means the model is doing well in both correctly detecting positive cases and reducing false alarms.

One of the main advantages of F1-score is that it works well when the dataset is imbalanced, such as in defect detection, where defective items are much fewer than non-defective ones. It is also useful when both types of errors—missing a defect or wrongly marking a good product as defective—are important to control. In such cases, F1-score helps give a fair evaluation by combining both aspects.

However, F1-score also has some limitations. Since it is a combined measure, it does not show the exact values of precision and recall separately. Two models with the same F1-score may behave very differently in real situations. Also, F1-score does not include the number of correctly predicted negative cases, so it doesn’t show the full picture. Because of this, it is recommended to use F1-score together with other metrics like accuracy for a better understanding of the model’s performance.

CHAPTER 3: CURRENT SITUATION ANALYSIS

3.1 Company introduction

3.1.1 Overview of Guoguang Electric Company Limited (GGEC)

Guoguang Electric Company Limited (GGEC), founded in 1951 in Guangzhou, China, is one of the leading and long-standing enterprises in the consumer electronics manufacturing industry. With over 70 years of development, GGEC has evolved into a comprehensive OEM and ODM service provider for globally recognized audio brands such as JBL, Harman/Kardon, BOSE, Sony, Amazon, and Panasonic.

The corporation employs over 6,000 staff and operates three major factories in China (Guangzhou, Huizhou, and Chongqing), one factory in Vietnam, and representative offices in the United States and Hong Kong. GGEC adopts international management systems including ISO 9001 for quality, ISO 14001 for environmental standards, and complies with global certifications for safety and sustainable manufacturing.

3.1.2 Guo Guang Electric Co., Ltd (Vietnam)

Guoguang Electric Company Limited (Vietnam) is a manufacturing subsidiary of GGEC, established and operating in the Dien Nam – Dien Ngoc Industrial Park in Quang Nam Province. This is the first GGEC plant located in Vietnam, built to expand production capacity and optimize the supply chain for international orders.

The company operates under a modern industrial model with specialized departments such as production, engineering, quality control, planning, logistics, and administration. The product range produced in Vietnam is similar to that of the group's main plants, including Bluetooth speakers, wireless earphones, control boards, and integrated electronic modules.

The production process at the plant is divided into distinct stages, including component reception, assembly, soldering, functional testing, visual inspection, and packaging. All stages are strictly monitored by the QA/QC department to meet both technical standards and aesthetic requirements from international clients.

Among the various stages, visual inspection is a critical step, still mainly carried out manually. Given the high aesthetic demands of the products, this step requires precision and stability in detecting small defects such as scratches, color mismatches, or misaligned components. This is why there is a need to improve the process through automation, particularly by applying machine vision and deep learning technologies to enhance the efficiency of product inspection.

3.2 Problem identification

The Charge 6 Bluetooth Speaker production began in mid-December 2024. Currently, the visual inspection process is experiencing longer inspection times and a higher defect rate than the standard threshold.

3.2.1 High defect rate issue

3.2.1.1 High defect rate

The visual inspection stage is essential for ensuring that the Charge 6 Bluetooth speaker meets appearance and quality standards before reaching customers. At this stage, trained inspectors carefully examine each unit for surface defects, assembly inconsistencies, and printing errors. This process helps identify defective products that do not meet quality specifications, preventing potential customer complaints and returns.

During the inspection process, several recurring defects have been observed, including scratched handle, scratched logo, dirty side cap, white spots on side cover, dirty IO port, scratched IO port cover, dirty rubber buttons, loose speaker grille

The defect rate threshold for this stage is set at 1.5%, meaning that ideally, no more than 1.5% of total units inspected should be classified as defective.

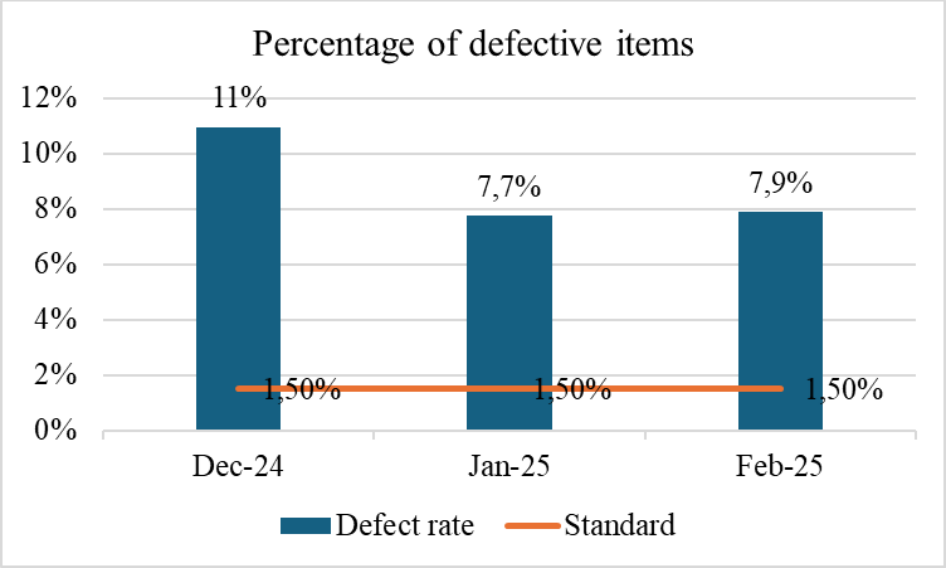


Figure 2. Defect rate by month

As shown in Figure 1, the defect rate at the visual inspection stage has consistently exceeded the 1.5% threshold, highlighting ongoing quality control issues. Specifically, in December 2024, the defect rate peaked at 11%, significantly surpassing the acceptable limit. Although some improvement was observed in January 2025, with the defect rate decreasing to 7.7%, it still remained well above the standard. In February

2025, the defect rate slightly increased again to 7.9%, indicating that the issue has not been fully resolved.

3.2.1.2 Measuring the high defect rate issue

During the visual inspection process, various defects can be identified in the Charge 6 Bluetooth Speaker. These defects impact both product quality and appearance, affecting customer perception and overall user experience. Below are definitions of the most common defects observed.

Table 1. Visual Inspection Defect Types

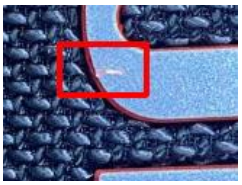
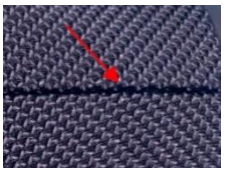
Defect type	Description	Defect Illustration
Scratches (side cover, IO cover, handle, PR, silicone buttons, logo)	Visible lines or marks on the surface that affect the appearance.	
Loose speaker grille	The grille is not securely attached, causing visible gaps.	
Misaligned passive radiators (PR)	The passive radiators are not positioned correctly or made from substandard materials.	
Dirty silicone buttons, grille, passive radiators, and IO section	These parts have dust, stains, or residue.	
White spots on side cover	White spots on side cover: Small stains or discolorations caused by material defects	

Table 2. Quantity and defect rate by defect type

	Dec-2024		**Jan-2025**		**Feb-2025**	
Input quanity	11.272		35.015		33.161	
Type of defect	**Number of defects**	**Defect rate**	**Number of defects**	**Defect rate**	**Number of defects**	**Defect rate**
Scratched logo	206	1,83%	402	1,15%	389	1,17%
Loose speaker grille	19	0,17%	52	0,15%	59	0,18%
Scratched IO cover	185	1,64%	373	1,07%	413	1,25%
Dirty IO cover	32	0,28%	89	0,25%	79	0,24%
Misaligned PR	14	0,12%	104	0,3%	119	0,36%
Scratched PR	88	0,78%	202	0,58%	216	0,65%
Dirty silicone buttons	37	0,33%	158	0,45%	137	0,41%
Scratched silicone buttons	234	2,08%	428	1,22%	380	1,15%
Scratched side cover	45	0,4%	117	0,33%	102	0,31%
White spots on side cover	11	0,1%	82	0,23%	79	0,24%
Scratched handle	339	3,01%	624	1,78%	571	1,72%
Dirty speaker grille	25	0,22%	78	0,22%	66	0,2%
Total	1.235	11%	2.709	7.7%	2.610	7.9%
Standard defect rate	1,5%		1,5%		1,5%	

3.2.1.3 Problem analysis

When the product reaches the visual inspection stage, it may encounter various defects caused by different factors during the production process. To analyze these causes more clearly, the Fishbone Diagram is used to identify and categorize the factors contributing to visual defects, as shown in Figure 2.

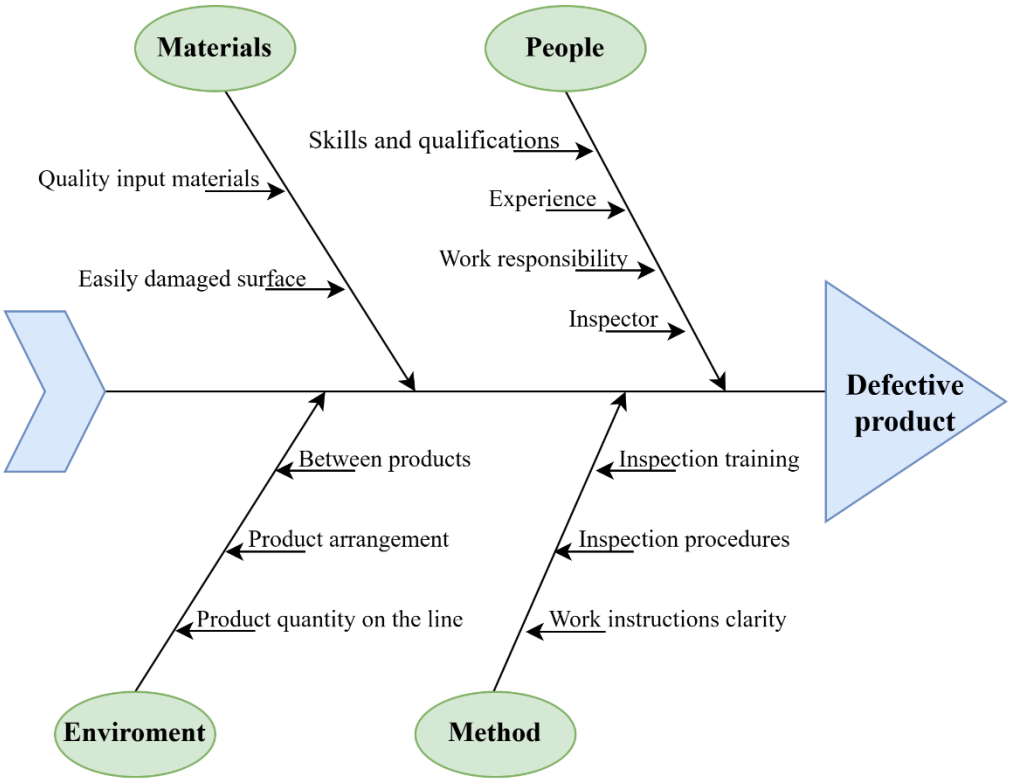


Figure 3. Fishbone Diagram of Visual Defect Causes

During the product assembly process, defects can arise from multiple sources, either due to a single cause or the cumulative effect of several factors. These factors can emerge at any stage of the production process, from raw material input, operational handling, and quality control to environmental conditions. Identifying and effectively controlling each cause will help minimize defects and enhance product quality.

A variety of defects commonly occur during the assembly process, as outlined in Table 4.1. Below are some key defect types and their potential causes:

Scratches: Surface scratches occur due to multiple factors during the production process. When products are placed too close together on the line, they may make contact, leading to unintended damage. Additionally, improper handling by workers during assembly increases the risk of contact with jigs, further contributing to scratches. Furthermore, low-quality materials are more prone to surface scratches.

Misalignment: Incorrect assembly orientation results from worker handling errors or unclear instructions.

Dirty: Raw materials arrive with dust or debris already present, or workstations and tools are not cleaned regularly, leading to dirt transferring onto products.

Loose speaker grille: The speaker grille is not fitted properly due to substandard materials. Insufficient or uneven assembly force prevents the grille from being securely fixed in place.

White spots on side cover: Poor-quality incoming materials result in white spots on the surface.

3.2.2 Issues with processing time

3.2.2.1 Extended inspection time

In the production process, the visual inspection stage is essential for ensuring product quality before moving to the next step. This stage is designed to detect defects such as scratches, dirt, misaligned assembly, or surface flaws to eliminate non-conforming products. However, if the inspection time exceeds the standard limit, it can disrupt the production flow, lead to product congestion on the assembly line, reduce overall efficiency, and increase operational costs.

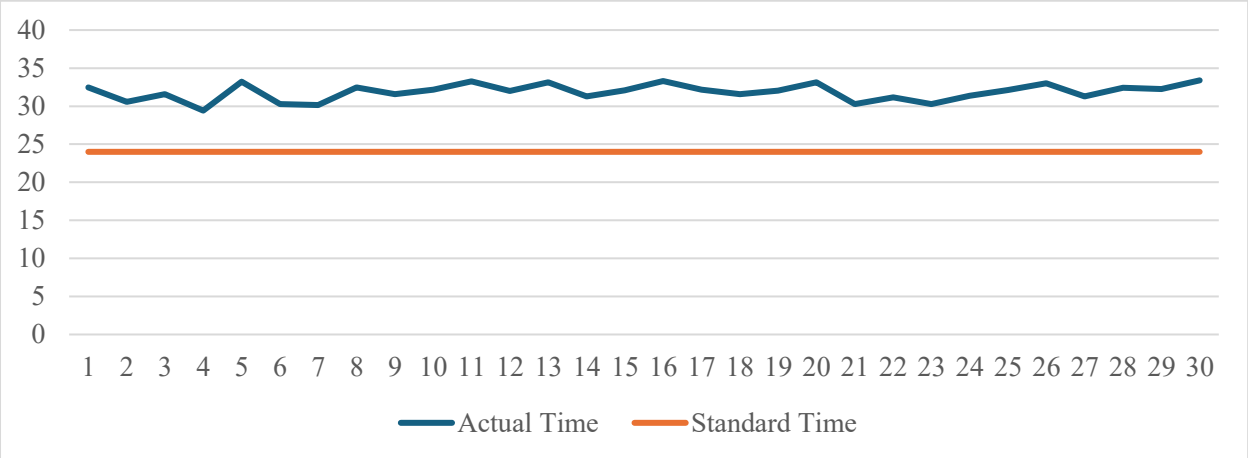


Figure 4. Actual vs. Standard Inspection Time

The chart above illustrates the comparison between actual time and standard time over 30 measurements. The standard time is fixed at 24 seconds, while the actual time ranges from 30 to 35 seconds.

It is clear that in all 30 measurements, the actual time is higher than the standard time. The actual time shows slight fluctuations but remains relatively stable, with an average of around 32 seconds, exceeding the standard by 6 to 11 seconds. This indicates

that the actual time consistently exceeds the standard time throughout the measurement process.

3.2.2.2 Problem analysis

In the production process, the visual inspection stage is essential to ensure that products meet quality standards before moving on to the next steps. However, the processing time for this stage is often longer than the standard time, affecting overall production efficiency.

Excessive time spent on visual inspection can lead to several consequences, such as:

- Product backlog on the production line, slowing down the production process.
- Increased labor costs, as inspection takes longer than expected.
- Reduced overall productivity, affecting the ability to meet order demands.

To better understand the causes of this issue, a Fishbone Diagram is used to analyze the factors contributing to prolonged visual inspection time. This diagram helps identify root causes from various aspects, including material, measurement, machine, method, environment, people.

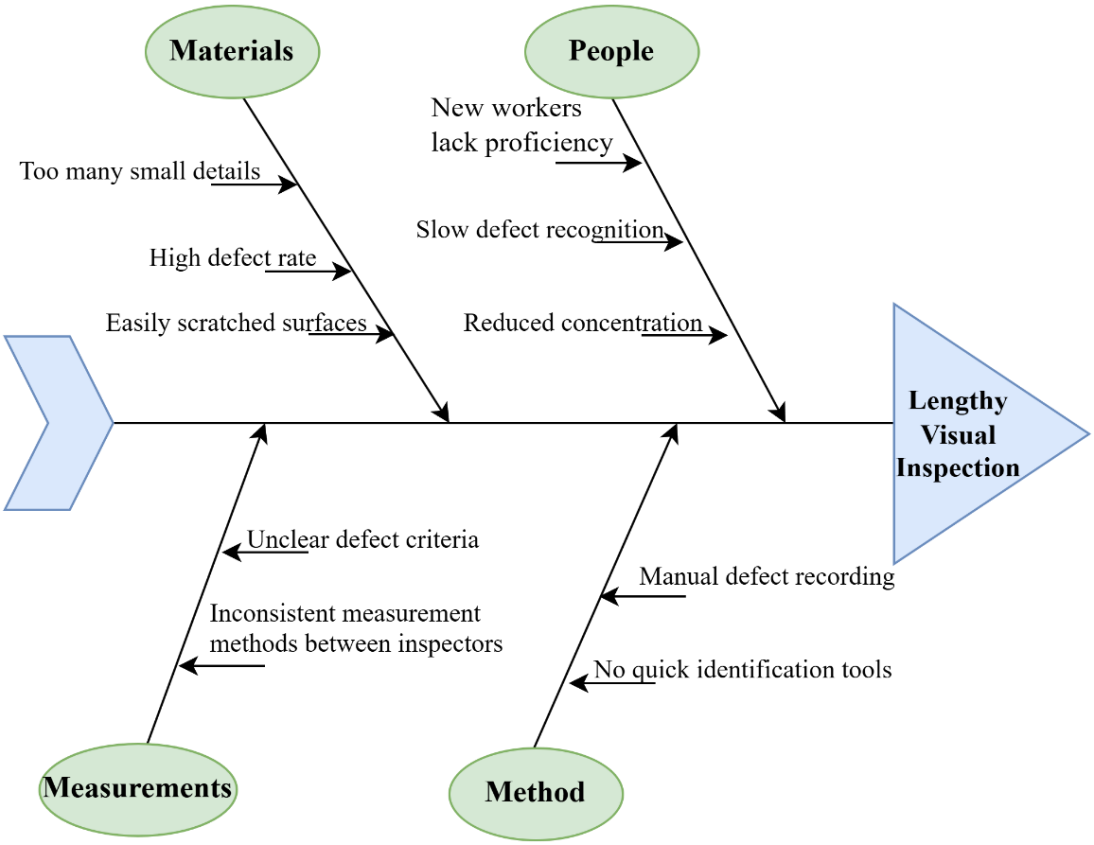


Figure 5. Causes of Lengthy Visual Inspection

Based on the diagram, there are four main causes affecting inspection time: People, Materials, Method, and Measurements. A detailed assessment of each group will help identify appropriate solutions to optimize the inspection process, minimize time waste, and improve production efficiency.

People: The skills and experience of workers also affect inspection time. New workers do not have enough experience, so they need more time to find and confirm defects. This makes the process slower. Slow defect recognition causes delays and uneven working speed. In addition, doing the same task for a long time reduces focus, leading to mistakes and longer inspection time.

Materials: The product has too many small details, increasing complexity in the inspection process. This forces employees to spend more time observing and identifying defects. Additionally, a high defect rate from the input stage significantly raises the number of products requiring inspection.

Moreover, easily scratched surfaces require more careful handling during inspection, limiting processing speed to avoid further damage to the product.

Method: Manual defect recording slows down the process as employees must write down defects by hand instead of using support tools. The lack of quick defect identification technology makes detection entirely dependent on visual inspection, reducing efficiency and increasing the risk of missed defects.

Measurements: Another contributing factor is the lack of clear and consistent measurement standards. Unclear defect criteria cause inspectors to spend extra time double-checking or consulting supervisors, slowing down the inspection process. Moreover, inconsistent measurement methods among inspectors lead to unreliable results and frequent re-checks, further extending inspection time and reducing efficiency.

3.2.3 Defect detection gaps

The quality control process is designed to ensure that products meet the required standards before reaching customers. By implementing multiple inspection stages, defects can be identified and corrected in a timly manner, preventing faulty products from being shipped.

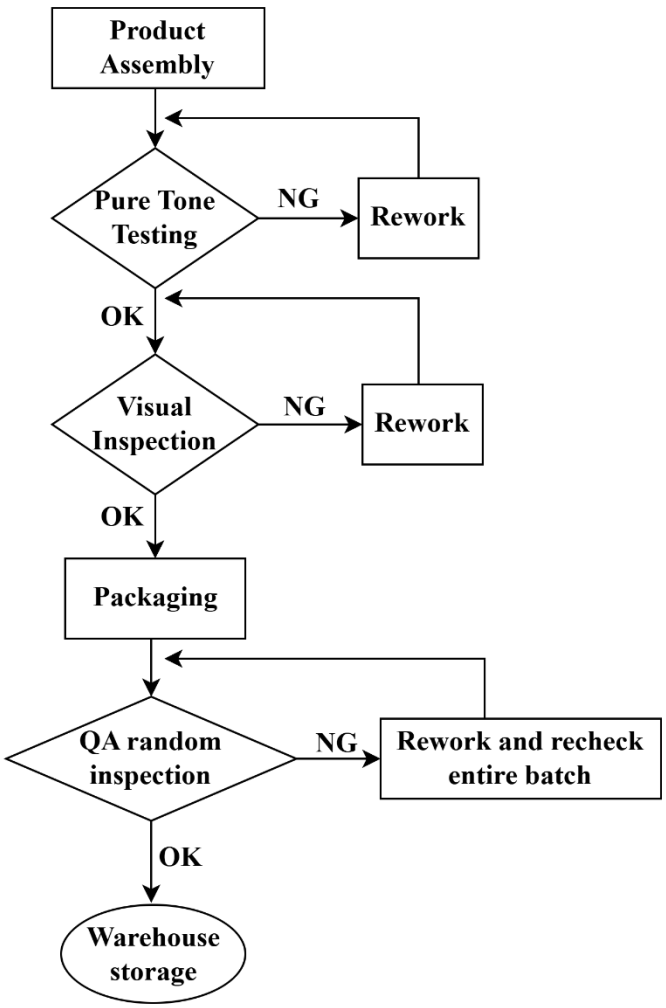


Figure 6. Quality control flowchart

After assembly, the components undergo inspection to ensure they meet quality standards. Once assembled, the product undergoes pure tone testing to check its sound quality. If any defects are found, the product is sent for rework; otherwise, it moves forward to the next stage.

Following this, a visual inspection is conducted to identify any external defects. If the product fails this inspection, it is reworked to correct the issues. Products that pass this stage proceed to the packaging step, where they are properly packed for storage and distribution.

After packaging, QA will conduct random inspections based on sample selection to ensure the overall quality of the entire batch before storage. If defects are detected during this stage, the entire batch is rechecked and reworked if necessary. If the sample meets quality standards, the products proceed to warehouse storage, marking the final step before shipment or distribution.

Despite undergoing a visual inspection process, some defective products are still detected during random sampling. This issue indicates gaps in the quality control system, where certain defects are overlooked, leading to inconsistencies in defect detection.

Table 3. Defect Inspection Report

Date	Inspection quantity	Sample size	Defect quantity	DPPM	Judgment result	Defective detection
01/06/2025	288	20	7	350000	Reject	3 pcs dirty PR
						2 pcs scratched IO
						2 pcs dirty IO
01/06/2025	288	20	6	300000	Reject	1pc dirty Logo
						1 pc misaligned PR
						2 pcs dirty speaker grille
						2pcs scratched IO
01/06/2025	288	24	3	125000	Reject	2 pcs scratched rubber button
						1 pc dirty speaker grille
01/07/2025	288	16	1	187500	Reject	1pc misaligned PR
01/07/2025	288	16	5	312500	Reject	5pcs dirty PR
01/08/2025	124	12	3	333333	Reject	1pc scratched IO
						1pc scratched Logo
						1 pc misaligned PR
01/08/2025	288	20	3	150000	Reject	1 pc dirty speaker grille
						2pcs chipped IO
01/09/2025	288	16	2	187500	Reject	1 pc dirty PR
						1 pc misaligned PR
01/10/2025	288	16	2	250000	Reject	2pcs loose speaker grille
01/11/2025	288	16	4	250000	Reject	2pcs scratched IO
						2pcs scratched logo

01/11/2025	288	20	5	250000	Reject	2pcs scratched IO 3pcs loose grille
01/13/2025	288	16	3	187500	Reject	2pcs scratched handles 1pc scratched logo
01/13/2025	288	24	3	208333	Reject	2 pcs misaligned PR 1 pc dirty speaker grille
01/13/2025	288	24	2	83333	Reject	1pc scratched rubber button 1pc scratched side cover
01/14/2025	288	36	3	83333	Reject	2 pcs white spots on side cover 1pc scratched IO
01/14/2025	288	20	4	200000	Reject	2pcs dirty PR 1pc scratched rubber button 1pc scratched side cover
01/17/2025	288	16	3	187500	Reject	1 pc dirty logo 2 pcs scratched IO
01/20/2025	288	16	4	250000	Reject	2 pcs scratched handle 1pc scratched logo 1pc scratched side cover
01/25/2025	288	16	2	125000	Reject	2 pcs chipped IO
02/10/2025	288	16	2	125000	Reject	2 pcs scratched logo
02/11/2025	288	16	3	187500	Reject	2 pcs scratched PR 1pc scratched side cover
02/12/2025	288	16	4	250000	Reject	3 pcs scratched handle 1pc scratched rubber button

Note:

- **pc:** piece
- **pcs:** pieces

In the table above, DPPM (Defective parts per million) is a key quality control metric that represents the number of defective parts found per one million inspected units. This measurement helps evaluate the overall defect rate in production and provides insights into process efficiency and product quality.

The formula used is:

$$\text{DPPM} = \left(\frac{Defect\ quantity}{Sample\ size}\right) * 1.000.000$$

In the given data, DPPM values vary depending on the number of defects detected in each sample. For example, on 01/06/2025, with a sample size of 20 and 7 defective units, the DPPM is 350,000. This indicates a high defect rate, leading to the rejection of the batch.

Table 3 provides an overview of the number of inspected products, sampled units, and detected defects. The data reveals shortcomings in the inspection process, as defects continue to be found even after prior quality checks. This suggests that the current inspection methods may not be entirely effective in identifying all defects, resulting in inconsistencies in product quality.

When defects are detected during random sampling, the entire batch is rejected. Consequently, the company must re-inspect the entire batch, leading to increased time consumption, higher labor costs, and potential delays in production.

❖ **Conclusion:**

From the aforementioned analysis, it is clear that the high number of defects overloads the inspection process, slowing down production progress. Manual defect detection relies entirely on human judgment, which not only takes more time but also increases the risk of errors. To enhance accuracy and efficiency, implementing an automated defect detection system is necessary. This system will help identify defects faster, reduce the workload on inspectors, and ensure more consistent product quality.

CHAPTER 4: CONVOLUTIONAL NEURAL NETWORK FOR AUTOMATED DEFECT DETECTION

This study focusses on comparing three popular CNN-based models: VGG16, YOLOv11, and InceptionV3 for classifying visual defects in Bluetooth speaker products. The reason for choosing these three models over other deep learning models stems from their specific advantages, which make them particularly well-suited for image classification tasks in production and quality control environments.

VGG16 is a relatively simple yet powerful convolutional neural network (CNN) model, excelling at image classification. Despite its straightforward architecture with convolutional layers and fully connected layers, VGG16 stands out in feature recognition, making it an excellent choice for basic to intermediate classification tasks, including visual defect detection.

YOLOv11 (You Only Look Once version 11), being a newer version of YOLO, offers the ability to detect objects quickly and accurately, especially when identifying and classifying visual defects across different regions of a product. With its real-time object detection capability, YOLOv11 is ideal for tasks that require high speed and precision in a production environment.

InceptionV3 is a more complex deep learning model that optimizes the learning and classification process. Thanks to its ability to learn complex features across multiple layers, InceptionV3 can achieve higher accuracy compared to simpler models like VGG16 in more complex classification tasks while still maintaining good performance.

We chose these three models not only for their effectiveness but also because they come with pre-trained versions on large datasets, saving time and effort in retraining from scratch. Comparing them will help identify which model performs best for classifying visual defects in Bluetooth speaker products while providing insights into speed, accuracy, and practical deployment.

4.1 Exploring YOLOv11 for Visual Defect Classification

4.1.1 Advancements and Key Features of YOLOv11

YOLOv11 represents a significant advancement in object detection technology, building upon the foundations laid by its predecessors, YOLOv9 and YOLOv10, which were introduced earlier in 2024. This latest iteration from Ultralytics showcases enhanced architectural designs, more sophisticated feature extraction techniques, and refined training methodologies. The synergy of YOLOv11's rapid processing, high accuracy, and computational efficiency positions are one of the most formidable models in Ultralytics' portfolio to date. A key strength of YOLOv11 lies in its refined

architecture, which facilitates the detection of subtle details even in challenging scenarios. The model's improved feature extraction capabilities allow it to identify and process a broader range of patterns and intricate elements within images. Compared to earlier versions, YOLOv11 introduces several notable enhancements:

- **Enhanced precision with reduced complexity:** The YOLOv11 variant achieves superior Mean Average Precision (mAP) scores on the COCO dataset while utilizing 22% fewer parameters than its YOLOv8m counterpart, demonstrating improved computational efficiency without compromising accuracy.
- **Versatility in CV tasks:** YOLOv11 exhibits proficiency across a diverse array of CV applications, including pose estimation, object recognition, image classification, instance segmentation, and oriented bounding box (OBB) detection.
- **Optimized speed and performance:** Through refined architectural designs and streamlined training pipelines, YOLOv11 achieves faster processing speeds while maintaining a balance between accuracy and computational efficiency.
- **Streamlined parameter count:** The reduction in parameters contributes to faster model performance without significantly impacting the overall accuracy of YOLOv11.
- **Advanced feature extraction:** YOLOv11 incorporates improvements in both its backbone and neck architecture, resulting in enhanced feature extraction capabilities and, consequently, more precise object detection.
- **Contextual adaptability:** YOLOv11 demonstrates versatility across various deployment scenarios, including cloud platforms, edge devices, and systems optimized for NVIDIA GPUs.

YOLOv11 model demonstrates significant advancements in both inference speed and accuracy compared to its predecessors. In the benchmark analysis, YOLOv11 was compared against several of its predecessors including variants such as YOLOv5 through to the more recent variants such as YOLOv10. As presented in Figure 6, YOLOv11 consistently outperforms these models, achieving superior mAP on the COCO dataset while maintaining a faster inference rate.

The performance comparison graph depicted in Figure 6 overs several key insights. The YOLOv11 variants (11n, 11s, 11m, and 11x) form a distinct performance frontier, with each model achieving higher COCO mAP^{50-95} scores at their respective latency points. Notably, the YOLOv11 achieves approximately 54.5% mAP^{50-95} at 13ms latency, surpassing all previous YOLO iterations. The intermediate variants, particularly

YOLOv11m, demonstrate exceptional efficiency by achieving comparable accuracy to larger models from previous generations while requiring significantly less processing time.

A particularly noteworthy observation is the performance leap in the low-latency regime (2-6ms), where YOLOv11 maintains high accuracy (approximately 47% mAP^{50-95}) while operating at speeds previously associated with much less accurate models. This represents a crucial advancement for real-time applications where both speed and accuracy are critical. The improvement curve of YOLOv11 also shows better scaling characteristics across its model variants, suggesting more efficient utilization of additional computational resources compared to previous generations.

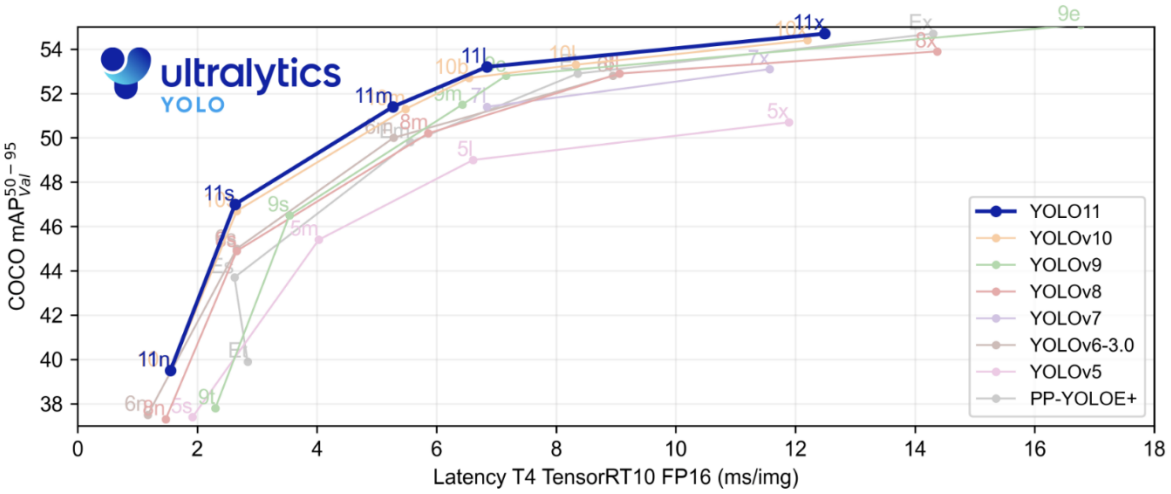


Figure 7. Benchmarking YOLOv11 Against Previous Versions [11]

4.1.2 YOLOv11 Model Architecture

The YOLOv11 architecture is designed based on the philosophy of "you only look once," which means that the model needs just one look at the entire scene to detect and classify objects. With its fast and accurate processing capabilities, YOLOv11 is used in defect classification problems, distinguishing between defect and non-defect products, by adapting the original detection architecture into a specialized structure for binary classification.

The entire network is divided into three main parts: Backbone – feature extraction; Neck – multi-scale feature aggregation and transmission; and Head – final classification.

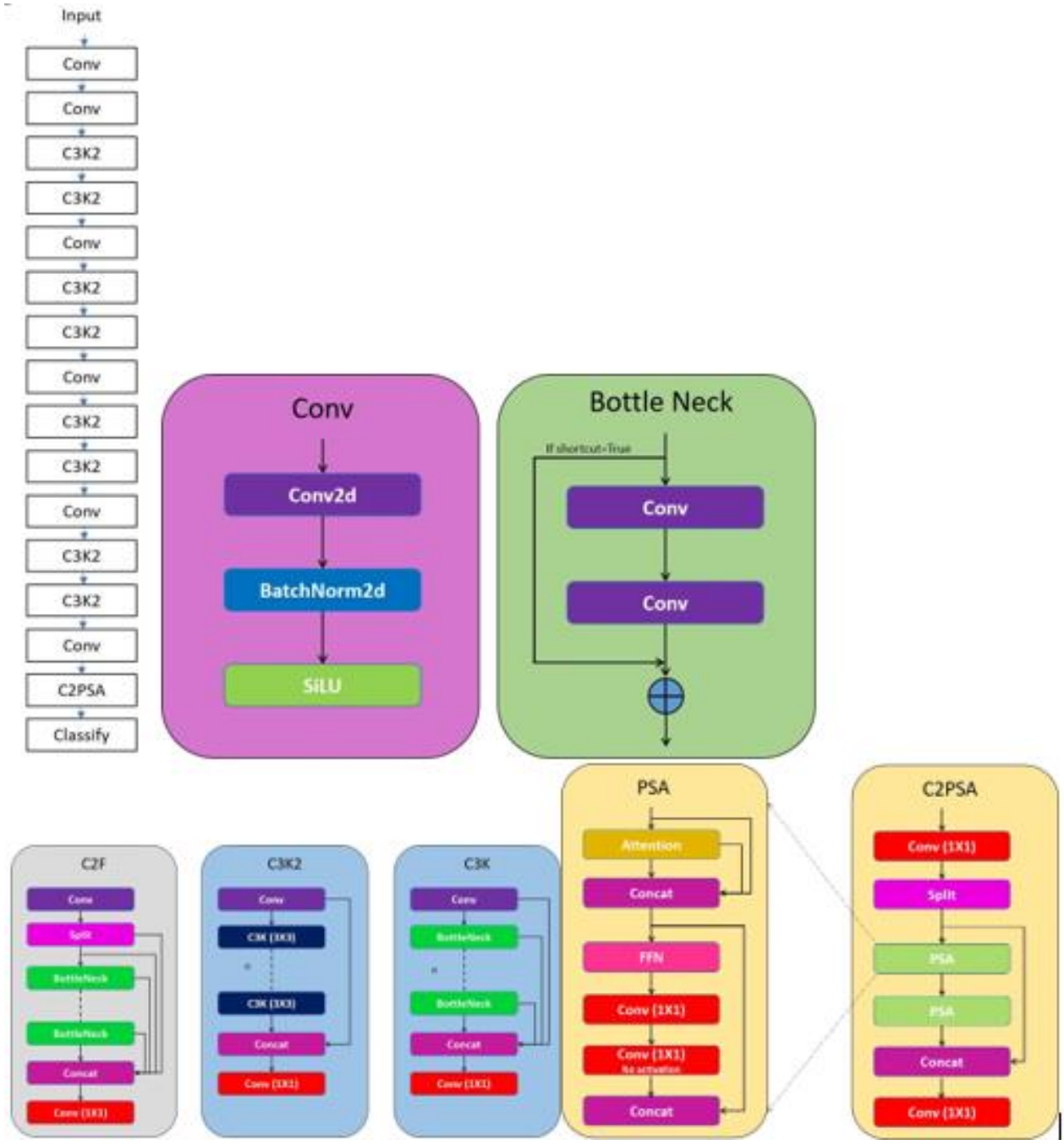


Figure 8. YOLOv11 Architecture Overview [12]

The input image, with a fixed size of 1760 × 1760 pixels, is passed through two consecutive Conv layers. Each Conv layer is composed of three elements: Conv2D, BatchNorm2D, and a SiLU activation function. These initial layers are responsible for denoising, normalization, and extracting low-level features such as edges, contours, and contrasts—essential for detecting surface-level defects.

Following this, the signal passes through two consecutive C3_k2 blocks. Each C3_k2 block includes two Bottleneck modules, designed based on the CSPNet principle,

allowing deep learning while preserving spatial information. After processing through these blocks, a Conv layer with stride = 2 is applied to downsample the spatial dimensions while enhancing the semantic representation.

This process repeats in a cyclical pattern: after each downsampling Conv layer, two C3_k2 blocks are applied. This results in a pyramid-like structure, where the spatial resolution decreases, but the depth of representation increases. Once this sequence is complete, the backbone concludes with a final Conv layer, passing the features to the next stage.

The extracted features are then fed into the Neck, which uses a C2PSA block to enhance spatial attention. The C2PSA block begins with a 1×1 Conv layer to reduce dimensionality, then splits into two branches passing through two PSA (Parallel Spatial Attention) modules. Within each PSA block, the model applies spatial attention, merges feature, processes them through a feed-forward network, and reconstructs the feature map using a 1×1 Conv layer. Finally, the two branches are combined using a Concatenate operation, ensuring that all significant features are preserved.

This attention mechanism allows the model to focus precisely on regions with a high likelihood of containing defects while filtering out irrelevant background noise. This enhances the ability to detect fine-grained or faint defects on product surfaces.

The final component, the Head, is responsible for performing the binary classification. After passing through the C2PSA block, the features are fed into a classification layer, which typically includes a Global Average Pooling layer and an output layer. The output is a single prediction indicating whether the product is defective or non-defective.

4.1.3 Pseudocode for Training YOLOv11 on Binary Image Classification Tasks

Pseudocode, also known as a "pseudo algorithm", is a way of representing an algorithm using human-readable language that is not bound by the strict syntax of any specific programming language. Instead of writing code in a particular language like Python or C++, pseudocode uses descriptive statements in Vietnamese or English, allowing readers to easily understand the main processing flow of a program.

The purpose of pseudocode is not for execution by computers, but rather to help people — including programmers, documentation readers, or other stakeholders — quickly and clearly grasp the logic and structure of the algorithm.

1. Begin
2. Load and preprocess dataset from Roboflow
3. Authenticate with Roboflow API

4. Download dataset version 6 in folder format
5. Rename validation folder from "valid" to "val"
6. Define image size: INPUT_SIZE = 480
7. Prepare training and validation data structure
8. Ensure structure: /train/class_name/*.jpg and /val/class_name/*.jpg
9. Load the YOLOv11 base model
10. input shape = (480, 480, 3)
11. pretrained weights = 'yolo11n-cls.pt'
12. Model automatically adapts to number of classes
13. Train the model
14. Call model. train () with the following parameters
15. Optimizer and learning rate
16. Loss function
17. Metrics: Top -1 accuracy
18. Evaluate performance on validation set
19. Call model.val (split = "val")
20. Generate confusion matrix and top-1 accuracy
21. Extract confusion matrix values
22. TP = m [0, 0]
23. FP = m [0, 1]
24. FN = m [1, 0]
25. TN = m [1, 1]
26. Precision = TP / (TP + FP)
27. Recall = TP / (TP + FN)
28. F1 Score = 2 * (Precision * Recall) / (Precision + Recall)
29. Accuracy = (TP + TN) / (TP + TN + FP + FN)
30. Predict class probabilities on validation set
31. Iterate through images in /val/ folder
32. Use model(image_path) to predict
33. Extract predicted class: probs. top1
34. Ground truth labels = 1 if "OK" in path, else 0
35. Plot ROC curve using sklearn
36. Compute FPR, TPR, thresholds = roc_curve (y_true, y_pred_probs)
37. AUC = auc (FPR, TPR)
38. Plot ROC curve with AUC label
39. Visualize training history
40. Display results.png and confusion_matrix.png

41. Plot training loss and accuracy over epochs if available
42. Evaluate final model
43. Predict class labels (threshold = 0.5)
44. Generate classification report
45. Convert model to TensorFlow Lite
46. Use model. export(format="tflite")
47. Save to file: yolo11n-cls. tflite
48. End

4.2 Exploring VGG16 for Visual Defect Classification

4.2.1 Introduction to VGG16 in Visual Defect Detection

In recent years, Convolutional Neural Networks (CNNs) have become a cornerstone of image classification tasks, particularly in applications involving visual quality inspection and defect detection. Among these networks, VGG16 stands out as a widely used and effective architecture due to its simplicity, depth, and strong performance on standard benchmarks such as ImageNet.

VGG16 follows a straightforward architectural design, stacking multiple 3×3 convolutional layers with ReLU activation, followed by max-pooling and fully connected layers. This homogeneous structure makes it easy to implement, tune, and transfer to new domains.

The model's ability to extract hierarchical features — from basic edges and textures to complex shapes — proves particularly useful in visual defect detection, where identifying small, subtle anomalies requires high-resolution feature representations.

Thanks to its pre-trained availability, VGG16 can be fine-tuned with a relatively small defect dataset, making it a practical choice for industrial applications that lack large-scale annotated data. This study investigates the effectiveness of VGG16 as a feature extractor and classifier for detecting visual defects on various product surfaces.

4.2.2 VGG16 Model Architecture

The VGG16 architecture, developed by the Visual Geometry Group at the University of Oxford, is a convolutional neural network (CNN) model renowned for its depth and uniform structure, making it effective for image classification tasks. The architecture consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. These layers are organized into blocks, with each block containing multiple convolutional layers followed by a max-pooling layer for downsampling.

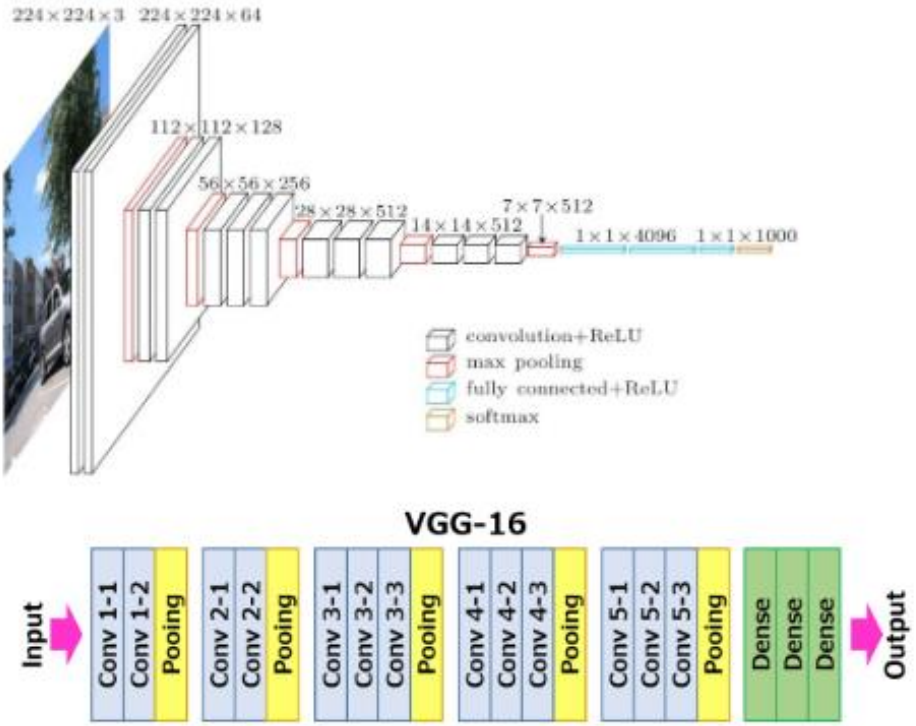


Figure 9. VGG-16 Architecture for Image Classification [13]

Table 4. Layer-wise Output Dimensions of VGG-16 Model

Layer (type)	**Output Shape**
input_layer (Input Layer)	(None, 224, 224, 3)
block1 conv1 (Conv2D)	(None, 224, 224, 64)
block 1_conv2 (Conv2D)	(None, 224, 224, 64)
block 1_pool1 (MaxPooling2D)	(None, 112, 112, 64)
block2_conv1 (Conv2D)	(None, 112, 112, 128)
block2_conv2 (Conv2D)	(None, 112, 112, 128)
block2_pool (MaxPooling2D)	(None, 56, 56, 128)
block3_conv1 (Conv2D)	(None, 56, 56, 256)
block3_conv2 (Conv2D)	(None, 56, 56, 256)
block3_conv3 (Conv2D)	(None, 56, 56, 256)
block3_pool (MaxPooling2D)	(None, 28, 28, 256)
block4_conv1 (Conv2D)	(None, 28, 28, 512)
block4_conv2 (Conv2D)	(None, 28, 28, 512)
block4_conv3 (Conv2D)	(None, 28, 28, 512)
block4_pool (MaxPooling2D)	(None, 14, 14, 512)
block5_conv1 (Conv2D)	(None, 14, 14, 512)
block5_conv2 (Conv2D)	(None, 14, 14, 512)
block5_conv3 (Conv2D)	(None, 14, 14, 512)
block5_pool (MaxPooling2D)	(None, 7, 7, 512)
flatten (Flatten)	(None, 25088)
dense (Dense)	(None, 512)

Layer (type)	Output Shape
dropout (Dropout)	(None, 512)
dense_1 (Dense)	(None, 1)

The input to any of the network configurations is considered to be a fixed size 224 x 224 image with three channels – R, G, and B. The only pre-processing done is normalizing the RGB values for every pixel. This is achieved by subtracting the mean value from every pixel.

Image is passed through the first stack of 2 convolution layers of the very small receptive size of 3 x 3, followed by ReLU activations. Each of these two layers contains 64 filters. The convolution stride is fixed at 1 pixel, and the padding is 1 pixel. This configuration preserves the spatial resolution, and the size of the output activation map is the same as the input image dimensions. The activation maps are then passed through spatial max pooling over a 2 x 2-pixel window, with a stride of 2 pixels. This halves the size of the activations. Thus, the size of the activations at the end of the first stack is 112 x 112 x 64.

The activations then flow through a similar second stack, but with 128 filters as against 64 in the first one. Consequently, the size after the second stack becomes 56 x 56 x 128. This is followed by the third stack with three convolutional layers and a max pool layer. The no. of filters applied here are 256, making the output size of the stack 28 x 28 x 256. This is followed by two stacks of three convolutional layers, with each containing 512 filters. The output at the end of both these stacks will be 7 x 7 x 512.

The stacks of convolutional layers are followed by three fully connected layers with a flattening layer in-between. The first two have 4,096 neurons each, and the last fully connected layer serves as the output layer and has 1,000 neurons corresponding to the 1,000 possible classes for the ImageNet dataset. The output layer is followed by the Softmax activation layer used for categorical classification.

4.2.3 Pseudocode for Training VGG16 on Binary Image Classification Tasks

1. Begin
2. Load and preprocess dataset from Roboflow
3. Authenticate with Roboflow API
4. Download dataset version 6 in folder format
5. Define image size: INPUT_SIZE = (224, 224)
6. Define batch size: BATCH_SIZE = 20
7. Create training and validation data generators
8. Apply rescaling: X' = X / 255
9. Use ImageDataGenerator for preprocessing
10. Load training and validation data from local directory

11. Load the VGG16 base model
12. input_shape = (224, 224, 3)
13. include_top = False
14. weights = 'imagenet'
15. Freeze all convolutional layers
16. Add custom classification layers
17. Flatten output from VGG16
18. Dense layer with 512 units, activation = ReLU
19. Formula: ReLU(x) = max (0, x)
20. Dropout layer with rate = 0.5
21. Dense output layer with 1 unit, activation = Sigmoid
22. Formula: Sigmoid(x) = 1 / (1 + exp(-x))
23. Compile the model
24. Optimizer: RMSprop with learning_rate = 0.001
25. Loss function: Binary cross-entropy
26. Formula: L (y, p) = -[y log(p) + (1 - y) log (1 - p)]
27. Metrics: Accuracy
28. Define training callbacks
29. EarlyStopping: monitor val_loss, patience = 20
30. ReduceLROnPlateau: monitor val_loss, patience = 5, factor = 0.3, min_lr = $1e^{-6}$
31. Custom callback: log learning rate per epoch
32. Train the model
33. Fit model on training data with validation
34. Epochs: 100
35. steps_per_epoch = floor (num_train_samples / BATCH_SIZE)
36. Evaluate performance
37. Predict probabilities on validation set: y_pred_probs
38. Get ground truth labels: y_true
39. Plot ROC curve using (FPR, TPR)
40. AUC = ∫ ROC curve
41. Visualize learning rate and training history
42. Plot learning rate across epochs
43. Plot training vs. validation loss and accuracy
44. Evaluate final model
45. Evaluate accuracy and loss on validation set
46. Predict class labels (threshold = 0.5)

47. Generate Confusion Matrix and Classification Report:
48. Precision = TP / (TP + FP)
49. Recall = TP / (TP + FN)
50. F1 Score = 2 * (Precision * Recall) / (Precision + Recall)
51. Convert model to TensorFlow Lite
52. Use TFLiteConverter
53. Apply optimization: tf.lite.Optimize.DEFAULT
54. Save to file: VGG16.tflite
55. End.

4.3 Exploring Inception V3 for Visual Defect Classification

4.3.1 Introduction to Inception V3 in Visual Defect Detection

Inception V3, an advanced version of the Inception model family, has achieved state-of-the-art results across many computer vision tasks. It introduces a range of architectural innovations that improve both computational efficiency and classification accuracy, making it well-suited for complex tasks such as defect detection.

Unlike traditional CNNs that follow a linear structure, Inception V3 uses Inception modules that perform multiple convolution operations in parallel, allowing the model to capture features at multiple scales. This ability is particularly beneficial for detecting defects of various sizes, shapes, and textures, which commonly occur in industrial settings.

Additionally, the model incorporates factorized convolutions and auxiliary classifiers to reduce the number of parameters and accelerate training without compromising performance.

Due to its robust feature extraction capacity and compatibility with transfer learning, Inception V3 offers a powerful alternative to conventional CNNs in visual defect detection. In this study, Inception V3 is explored to assess its capability in identifying subtle visual anomalies and enhancing the overall accuracy of the classification system.

4.3.2 Inception V3 Model Architecture

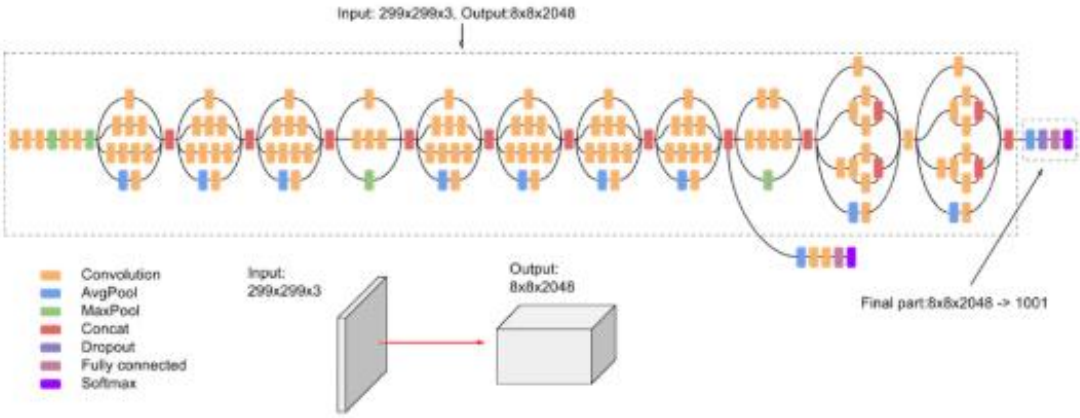


Figure 10. Inception V3 Model Architecture [14]

The input to Inception V3 is an RGB image of size 299×299×3. Before entering the network, the image is normalized to ensure consistency and stability in training.

At the core of Inception V3 are multiple **Inception modules**, each consisting of parallel convolutional paths using different kernel sizes (1×1, 3×3, 5×5). These parallel paths enable the model to capture image features at various scales simultaneously.

To reduce the number of parameters and improve computational efficiency, factorized convolutions are applied. For instance, a 5×5 convolution is replaced by two consecutive 3×3 convolutions. Furthermore, asymmetric convolutions are introduced by replacing a 3×3 convolution with a 1×3 followed by a 3×1 convolution. These changes significantly reduce computation while preserving performance.

Between Inception blocks, MaxPooling and AveragePooling layers are inserted to reduce the spatial dimensions of feature maps and control overfitting. Additionally, grid size reduction techniques are employed to efficiently reduce the resolution of the feature maps without losing essential information. This contributes to both model depth and training stability.

To prevent the vanishing gradient problem and improve learning in deeper networks, auxiliary classifiers are embedded within intermediate layers. These classifiers act as additional supervision during training and serve as regularizers, enhancing model generalization and convergence.

After processing through all Inception blocks and pooling layers, the final feature map is flattened and passed through fully connected layers. The last layer is a **softmax**

layer that produces a probability distribution of over 1000 classes, typically for classification tasks on the ImageNet dataset.

4.3.3 Pseudocode for Training Inception V3 on Binary Image Classification Tasks

1. Begin
2. Load and preprocess image data
3. Use ImageDataGenerator to normalize pixel values:
4. For each pixel: X' = X / 255
5. Set batch size B = 20
6. Resize images to shape (150, 150, 3)
7. Load pre-trained InceptionV3 model
8. Input shape: (150, 150, 3)
9. Include_top = False (remove default classifier)
10. Weights = 'imagenet'
11. Freeze all layers to avoid updating pre-trained weights
12. Add custom classifier layers
13. Flatten output of base model
14. Dense layer: 1024 units, activation = ReLU
15. Formula: ReLU(x) = max (0, x)
16. Dropout layer with rate = 0.2
17. Output layer: 1 unit, activation = Sigmoid
18. Formula: Sigmoid(x) = 1 / (1 + exp(-x))
19. Compile the model
20. Optimizer: RMSprop with learning rate $\alpha = 0.001$
21. Loss function: Binary Crossentropy
22. Formula:
23. L (y, p) = - [y * log(p) + (1 - y) * log (1 - p)]
24. where:
25. $y \in \{0,1\}$ is true label
26. $p \in [0,1]$ is predicted probability
27. Metric: Accuracy
28. Formula: Accuracy = (TP + TN) / (TP + TN + FP + FN)
29. Configure training callbacks
30. EarlyStopping: monitor val_loss, patience = 20
31. ReduceLROnPlateau:
32. if no improvement in val_loss for 5 epochs:
33. learning_rate = learning_rate × 0.3

34. min_learning_rate = $1e^{-6}$
35. Train the model
36. Epochs: up to 100
37. Steps per epoch = floor (N / B)
38. Validation data evaluated after each epoch
39. Evaluating model performance
40. Calculate loss and accuracy on validation set
41. Make predictions: $p \in [0,1]$
42. If $p \geq 0.5 \rightarrow$ Predict class 1
43. Else $\rightarrow$ Predict class 0
44. Generate evaluation metrics
45. Confusion Matrix:
46. [[TP, FN],
47. [FP, TN]]
48. Classification Report: including Precision, Recall, F1
49. Formula:
50. Precision = TP / (TP + FP)
51. Recall = TP / (TP + FN)
52. F1 Score = 2 * (Precision * Recall) / (Precision + Recall)
53. ROC Curve: plot (FPR, TPR)
54. where:
55. FPR = FP / (FP + TN)
56. TPR = TP / (TP + FN)
57. AUC = $\int$ ROC curve area
58. Convert model to TensorFlow Lite
59. Apply quantization optimization
60. Save model as `InceptionV3.tflite`
61. End.

4.4 Data Collection

4.4.1 Objective

The primary objective of the data collection process in this project was to build a comprehensive and high-quality dataset for training a classification model capable of detecting subtle defects in Bluetooth speakers. The defects targeted are often small, difficult to identify with the naked eye, and lack clear boundaries. Therefore, the data

collectionO phase required careful planning to ensure that the dataset is representative, diverse, and suitable for training a deep learning model to perform reliably in real-world conditions.

4.4.2 Sample Sources

The training dataset was collected directly from the visual inspection stage of the production line during the quality control process. Sampling real-world conditions ensures that the dataset accurately represents potential defects in Bluetooth speakers. The defects, as defined in Table 1, include scratches, loose speaker grilles, misaligned passive radiators, dirty silicone buttons, grilles, passive radiators, IO sections, and white spots on the speaker cover. These defects are often small, randomly distributed, and difficult to detect with the naked eye. Therefore, high-quality image data is essential for training the model to recognize these issues effectively.

To maintain consistency and optimize model training, data collection followed specific guidelines:

- Stable lighting conditions: Uniform lighting was maintained to prevent shadows and bright fluctuations, ensuring clearer defect detection.
- Fixed camera angles: Standardized angles were used to maintain consistency and minimize variations in defect detection.
- High-resolution images: Sufficient resolution was ensured to capture fine details, enabling the model to detect even minor defects.
- Complete product coverage:
    - For OK products (defect-free units), images were captured from six different sides (top, bottom, front, back, left, and right) to provide comprehensive training data.
    - For NG products (defective units), images were taken of the entire side containing the defect, rather than just focusing on the specific defective area. This approach helps the model recognize defects in different positions and under various conditions.
- No foreign objects: Images were captured in a controlled environment to prevent unwanted elements from interfering with model training.
- No cropped sections: The entire product was included in each frame to avoid missing critical areas.

By adhering to these structured data collection methods, the dataset remains high-quality, ensuring the deep learning model performs accurately and consistently when deployed in real-world applications.

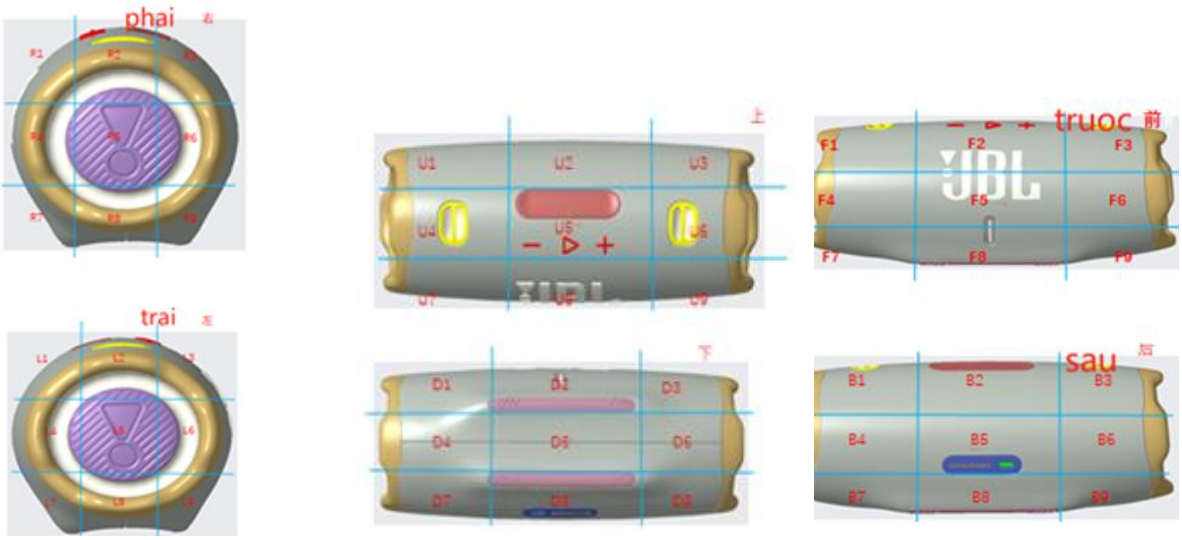


Figure 11. Six-Side View of Bluetooth Speaker

4.5 Data processing with Roboflow

In the process of building a deep learning model, image data processing plays a crucial role in ensuring high-quality input, thereby improving the model's recognition performance. To achieve this, data needs to be standardized and augmented to help the model learn diverse features and enhance generalization to real-world data. In this study, Roboflow is utilized as a comprehensive platform to handle data preprocessing, dataset splitting, and data augmentation. These steps ensure that the input data is well-structured and optimized for training a high-performance model.

4.5.1 Data Preprocessing

Data preprocessing is the initial step in standardizing images to ensure consistency and optimize machine learning performance. The preprocessing techniques applied include:

- **Auto-Orient: Applied**

One common issue with image datasets is inconsistent orientation due to metadata stored by camera devices. Some images may be rotated incorrectly depending on the device used to capture them. Roboflow automatically corrects the orientation of all images to a standardized format. This prevents misalignment issues during training and ensures that the model learns consistent patterns across all samples.

- **Resize: Stretch to 1760×1760 pixels**

Deep learning models require fixed input dimensions to function effectively. In this study, all images are resized to 1760×1760 pixels using a stretching method. Unlike

cropping, which may remove valuable information from the image, stretching ensures that the entire image remains intact. Resizing also helps optimize processing speed by ensuring uniform input sizes, making computations more efficient during training.

4.5.2 Dataset Splitting

To ensure the model is trained effectively and evaluated fairly, the dataset used in this study was divided into two main subsets: 70% for training and 30% for validation. This method of splitting is commonly applied in machine learning tasks to provide a good balance between learning and evaluation**.**

The training set (70%) serves as the core data source for the model to learn from. It contains the majority of labeled samples, enabling the model to identify patterns, features, and relationships that are critical for classification tasks.

Meanwhile, the validation set (30%) plays an essential role in monitoring the model's performance during training. It is not used to update the model's weights but rather helps to evaluate how well the model generalizes to unseen data. This aids in early detection of overfitting, supports hyperparameter tuning, and assists in selecting the best model version.

The choice of a 70:30 ratio allows for a sufficient volume of data to optimize the model's learning while still preserving a significant portion for validation, ensuring that the model's performance metrics are reliable and unbiased.

Table 5. OK/NG Dataset Split

Dataset	Number of samples	Training set (70%)	Validation set (30%)
With defects (NG)	987	691	296
Without defects (OK)	1021	715	306
Total	2008	1406	602

4.6 Two-Stage Classification Framework

This project proposes a two-stage classification framework to enhance the accuracy and efficiency of visual defect inspection for Bluetooth speakers.

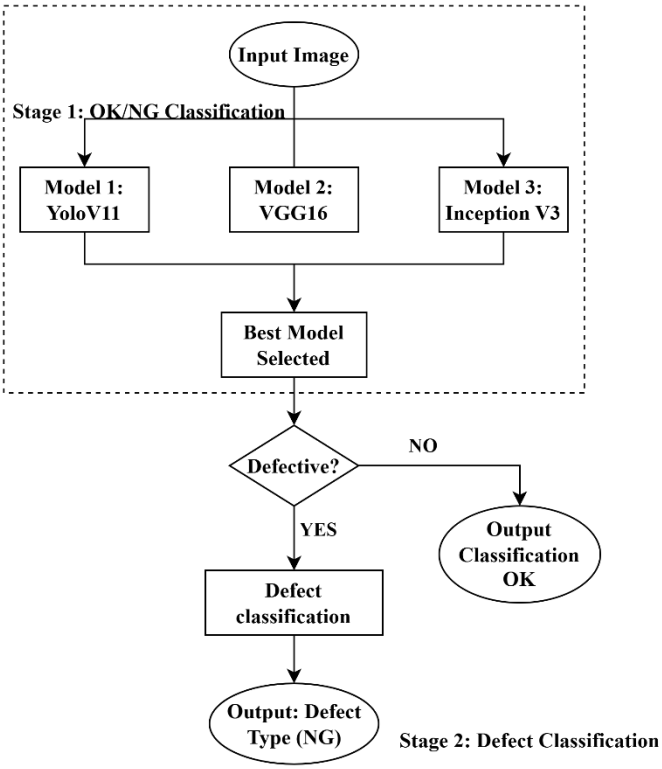


Figure 12.Two-Stage Classification Process for Visual Inspection

In the first stage, three deep learning models including YOLOv11, InceptionV3, and VGG16 are trained to classify products into two categories: OK for non defective products and NG for defective ones. The main goal of this stage is to determine which model performs best based on training stability, loss, and classification accuracy. The model with the best overall performance is selected to move on to the second stage.

In the second stage, the chosen model is retrained to focus on identifying specific types of visual defects found in NG products. In the second stage, the chosen model is retrained to focus on identifying the specific locations of visual defects found in NG products. These defects typically occur at key areas such as the handle, IO cover, logo area, or silicone buttons. By dividing the process into two clear stages, the system achieves better detection accuracy and clearer classification of defects, which supports practical deployment in manufacturing environments.

CHAPTER 5: MODEL PERFORMANCE ANALYSIS

5.1 Training Process Analysis

During the model training process, the computational environment and programming tools play a critical role in ensuring system performance and training stability. In this study, all training experiments were conducted on Kaggle, a widely used cloud-based platform that provides free GPU resources optimized for deep learning tasks. To maximize computational efficiency and reduce training time, the system was configured to utilize 2 NVIDIA Tesla T4 GPUs, enabling fast parallel processing of tensor operations and improved backpropagation speed.

The primary programming language used throughout the project was Python 3, along with commonly used deep learning libraries such as TensorFlow, Keras, PyTorch, OpenCV, and Scikit-learn. These libraries supported key stages of the workflow, including data preprocessing, model architecture development, training, and performance evaluation.

A key training hyperparameter - learning rate - was initially set to 0.01. Following a learning rate decay strategy, this value was gradually reduced throughout the training process to ensure stable convergence and prevent overshooting the local minima. By the final stages of training, the learning rate was decreased to 10^{-7}, allowing for fine-tuned weight adjustments and better generalization.

5.1.1 Learning Process Evaluation of YOLOv11

The figure below illustrates the training process of the YOLOv11 model through four key visualizations: train loss, validation loss, Top - 1 accuracy, and Top -5 accuracy. These charts reflect the evolution of training metrics across epochs and serve as a foundation to assess the model's learning ability, convergence behavior, and training stability throughout the process**.**

a. Train Loss

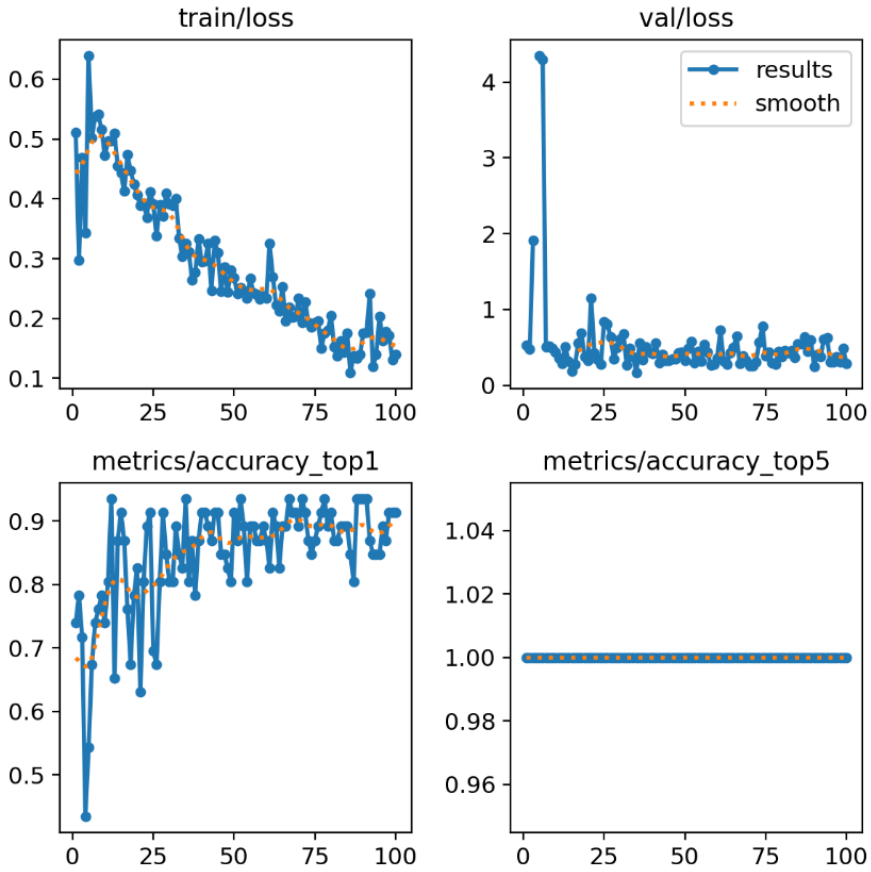


Figure 13. OK/NG Model Training Curves

The train/loss chart reflects the variation of the loss function value during the model's training on the training dataset.

The loss function is a critical component in deep learning, used to measure the discrepancy between the model's prediction and the actual values. During training, the optimization algorithm aims to minimize this loss value as much as possible. Monitoring the train loss across epochs shows whether the model is learning effectively. A well-trained model typically has a steadily decreasing loss curve approaching a low value.

Observing the train/loss chart in the figure reveals that the initial loss value starts at approximately 0.6, which is relatively high and reflects the model's untrained state, having not yet learned any features from the data. However, after just a few initial epochs, the loss begins to decrease steadily, indicating that the model is gradually assimilating information from the training data and starting to learn fundamental patterns.

From approximately epoch 10 to epoch 50, the loss continues to decline at a stable rate, fluctuating around 0,3 to 0,2. This downward trend is a positive indicator of an effective learning process, as the model's predictions increasingly approximate the actual labels. The loss curve is notably smooth, accompanied by a dashed line representing a smoothed moving average, which helps highlight the overall trend more clearly.

Toward the final phase of training, between epoch 70 and epoch 100, the training loss further decreases to around 0,13 to 0,15, suggesting that the model is approaching an optimal state on the training set. Although minor fluctuations are observed, the overall trend continues downward, demonstrating strong convergence and training stability.

b. Validation Loss

The val/loss chart illustrates the variation of the loss value on the validation dataset over training epochs. Unlike training loss, which is calculated on data the model has seen, validation loss is computed on unseen data, making it a key metric to evaluate the model's generalization ability. It indicates how well the model performs on new, untrained data and helps identify potential overfitting issues.

From the chart, it is observed that during the initial epochs, the validation loss is relatively high (around 4,2), indicating limited generalization performance at the early stage of training. However, after approximately 10 epochs, the validation loss drops sharply and then begins to stabilize. Between epoch 20 and 100, the loss fluctuates within the range of 0,3 to 0,5; suggesting that the model gradually learns meaningful features that generalize well to unseen data. Although some minor fluctuations remain, the overall trend does not reverse, which is a positive sign showing that the model is not suffering from significant overfitting.

c. Evaluation of Top -1 and Top - 5 Accuracy

In evaluating the performance of deep learning models, accuracy metrics are used to reflect the model's ability to make correct predictions. In this study, two key metrics are monitored: Top - 1 Accuracy and Top - 5 Accuracy, which assess the model's performance in classifying products into two categories: defective and non-defective.

- **Top – 1 Accuracy**

Top - 1 Accuracy represents the percentage of samples where the model's top prediction matches the true label. In the binary classification task of defect detection, this metric directly reflects the model's classification precision and is considered the most important indicator of real-world performance.

A high and stable Top - 1 Accuracy indicates that the model can clearly distinguish between defective and non-defective products, ensuring reliability when deployed in automated visual inspection systems. The accuracy trend observed during training shows a steady increase, reaching above 90% after around 60 epochs, suggesting an efficient and stable learning process.

❖ **Top 5 – Accuracy**

Although Top - 5 Accuracy is typically used in multi-class classification problems, it is also monitored in this study to observe the model's confidence and label ranking consistency. In a binary classification context, the Top - 5 Accuracy is almost always at or nearly 100%. Nevertheless, the fact that the model consistently includes the correct label within its top predictions demonstrates its robustness in making clear decisions between two classes.

5.1.2 Learning Process Evaluation of VGG16

a. Learning Rate

The chart below illustrates the variation of the learning rate parameter throughout the training process of the VGG16 model. This parameter plays a critical role, as it directly impacts the speed and effectiveness of weight optimization. Specifically, the graph depicts how the learning rate changes over epochs, reflecting the learning rate scheduling strategy applied during training.

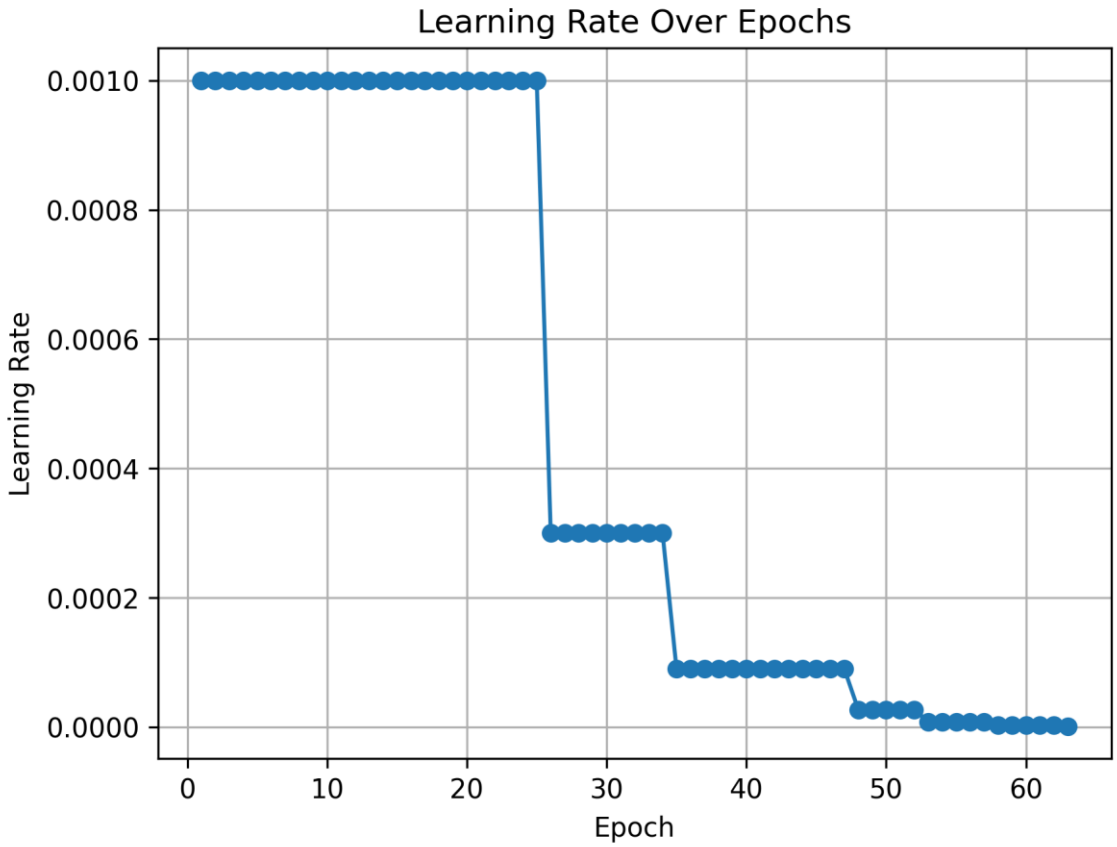


Figure 14. VGG16 Learning Rate Over Epochs

During the first 25 epochs, the learning rate was kept constant at **0,001**. This is relatively high value, suitable for allowing the model to quickly learn basic features from the input data. Maintaining a high learning rate in this stage accelerates convergence and helps prevent the model from getting stuck in local minima.

After Epoch 25, the learning rate was sharply reduced to **0,0003**. This adjustment is made to slow down weight updates as the model begins to learn more complex patterns. Lowering the learning rate at this stage helps the training process become more stable and less prone to oscillations.

The learning rate is further decreased to **0,0001**, allowing the model to fine-tune the parameters it has learned. This phase is crucial for improving accuracy and reducing loss with more precise updates, minimizing drastic changes in the weight values.

From epoch 48 onward, the learning rate continues to decrease to very small values such as 0,00003; 0,00001 and gradually approaches 0. The objective here is to make subtle refinements, ensuring the model achieves optimal performance before completing training.

The strategy of gradually reducing the learning rate over time is a highly effective and widely adopted technique in deep neural network training. This approach enables the model to learn quickly at first, stabilize during mid-training, and reach fine-tuned optimization at the end. As a result, the VGG16 model is better able to converge, avoid overfitting, and improve classification accuracy.

b. Loss Curve

In training convolutional neural networks (CNNs), tracking the training history through key metrics such as loss and accuracy is essential for evaluating how well the model learns. These metrics are typically computed and visualized for both the training set and the validation set, helping assess the model's learning progress, convergence behavior, and generalization ability on unseen data.

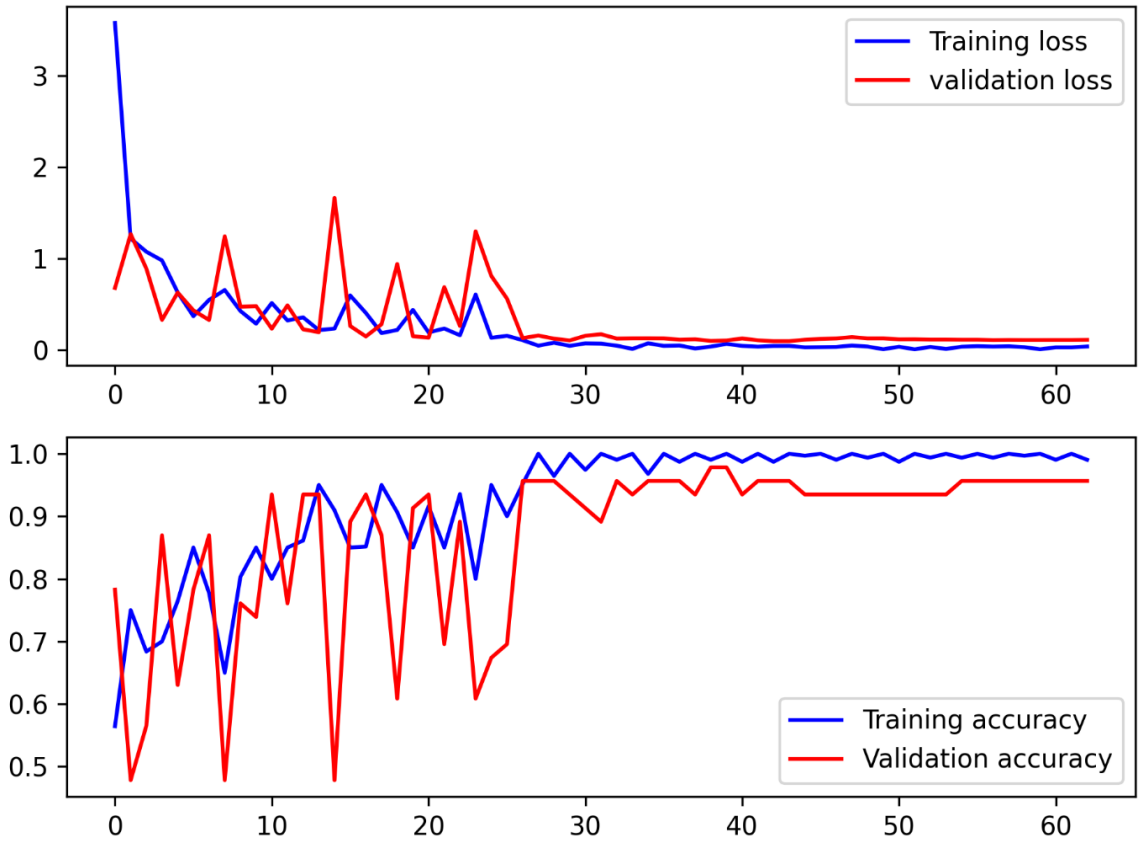


Figure 15. VGG16 Loss and Accuracy Curves

Based on the training history chart shown in Figure 13, this section analyzes the learning process of the VGG16 model through two key evaluation metrics: loss and accuracy, recorded on both the training and validation sets over a total of 60 epochs.

- The upper graph shows the loss, which measures the difference between the predicted output and the actual label.

- The lower graph shows the accuracy, which indicates the proportion of correct predictions made by the model.

As shown in the upper part of Figure 13, the training loss starts at a relatively high level (above 3.0), indicating that the model initially has not yet learned from the data. However, the loss decreases rapidly during the first 10 epochs, reflecting an effective initial learning phase.

After approximately the 10th epoch, the training loss continues to decline steadily and stabilizes below **0.2** by around epoch 30, suggesting that the model has successfully converged.

Meanwhile, the validation loss also exhibits a downward trend, with slight fluctuations—this is common when evaluating unseen data. Importantly, the validation loss does not increase in the later stages of training, indicating no significant overfitting. The narrowing gap between training and validation loss toward the end of training further confirms consistent and balanced model learning.

c. Accuracy Curve

In the lower part of Figure 13, the accuracy chart shows that training accuracy increases rapidly from around 0.6 to over 0.95 within the first 20 epochs and remains stable at a high level thereafter. This demonstrates that the model effectively learns from the training dataset.

The validation accuracy also improves over time, despite slight fluctuations in the early stages. Beginning around Epoch 30, the validation accuracy gradually aligns with the training accuracy and reaches comparable levels (above 0.95) in the final epochs. This close alignment indicates that the model not only performs well on the training data but also generalizes effectively new, unseen inputs.

The convergence of the accuracy curves is a strong indicator of a balanced training process and supports the model's reliability for deployment in practical defect detection scenarios.

5.1.3 Learning Process Evaluation of Inception V3

a. Learning Rate

The chart below illustrates the change in learning rate throughout the training process of the InceptionV3 model. The learning rate is a crucial hyperparameter that controls how much the model weights are updated during backpropagation. Adjusting the learning rate at different training stages helps optimize learning efficiency and ensures proper model convergence.

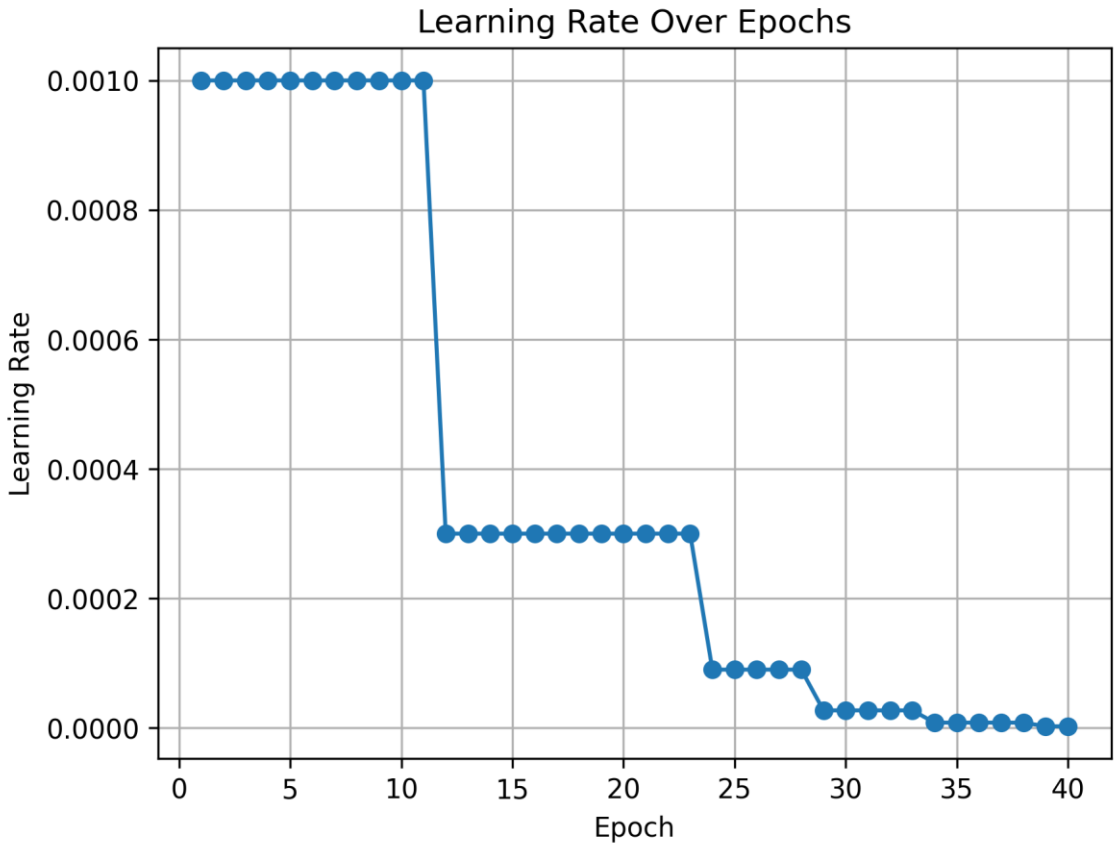


Figure 16. Learning Rate over Epochs for Inception V3

From epoch 0 to epoch 11, the learning rate is kept constant at **0.001**. During this phase, the model learns quickly, focusing on exploring the parameter space and building basic feature representations. A high learning rate allows the model to learn faster while maintaining training stability in the early stage.

At epoch 12, the learning rate drops significantly to 0.0003. This adjustment slows down the update process, allowing the model to stabilize and begin fine-tuning deeper features. A lower learning rate helps reduce oscillations and prevents overfitting.

The learning rate continues to decrease to 0.0001 from epoch 23 to epoch 27, a common value used for fine-tuning. At this point, the model aims to refine its parameters more precisely to improve performance on the validation set.

From epoch 28 to epoch 40, the learning rate continues to decline gradually, reaching very small values around 0.00001, allowing the model to fine-tune its parameters as precisely as possible to enhance validation performance.

b. Loss Curve

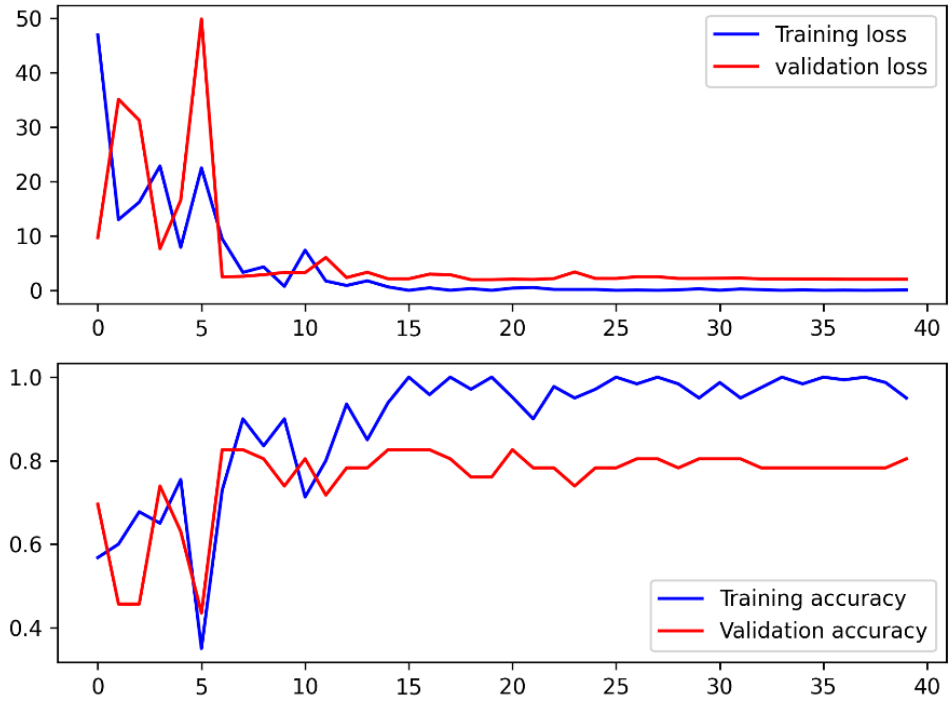


Figure 17. Inception V3 Loss Curve

Based on Figure 15, the training process of the InceptionV3 model can be evaluated through two main graphs: the loss graph and the accuracy graph, which show the performance on both the training and validation sets.

As observed in the upper part of Figure 15, during the initial stage (from epoch 0 to approximately epoch 7), both training loss and validation loss exhibit significant fluctuations. This behavior is commonly observed during the early optimization phase, where the model is still adjusting its parameters and has not yet effectively learned from the data.

Starting around Epoch 7, the model becomes more stable, with the training loss decreasing rapidly and remaining at a low level starting around Epoch 15. This indicates that the model has effectively learned the classification features from the training data. However, the validation loss continues to fluctuate and does not decrease as noticeably as the training loss. This discrepancy suggests the model begins to show signs of overfitting, meaning it has started to memorize the training data rather than generalizing well to unseen data.

c. Accuracy Curve

In the lower part of Figure 15, the training accuracy shows a sharp upward trend, surpassing 0.95 by approximately epoch 15, and then maintaining this high level of

performance. This reflects the model's ability to capture key discriminative features within the training set.

In contrast, the validation accuracy stabilizes around 0.8, with no significant improvement observed in the subsequent epochs. The noticeable gap between the training and validation accuracy once again points to the issue of overfitting, highlighting that while the model performs exceptionally on known data, its generalization capability remains limited.

5.2 Comparative Evaluation of Models

5.2.1 Comparison of Classification Metrics

Table 6 presents the performance evaluation results of three deep learning models: YOLOv11, VGG16, and InceptionV3. The evaluation is based on four widely used classification metrics: Accuracy, Precision, Recall, and F1-Score. These indicators not only reflect the overall correctness of the model's predictions but also represent the balance between the model's ability to detect actual defects and avoid false alarms.

Table 6. Classification Performance Comparison

Metric	Accuracy	Precision	Recall	F1-Score	Time (s)
YOLOv11	0,93	0,93	0,94	0,93	20
VGG16	0,96	0,96	0,96	0,96	7
InceptionV3	0,76	0,76	0,77	0,76	9

The results show that VGG16 achieved the highest overall performance, with all four metrics reaching 0.96. This indicates that the model effectively learns discriminative features between the two classes and maintains high prediction stability. The consistency between Precision and Recall demonstrates a strong balance between detecting defective products and minimizing false positives, which is crucial for ensuring the reliability of an automated inspection system. These strengths affirm that VGG16 is a highly suitable candidate for real-world quality control applications.

In comparison, YOLOv11 also demonstrated promising results, with Accuracy, Precision, and F1-Score all at 0.93, and Recall slightly higher at 0.94. The higher Recall suggests that YOLOv11 leans toward more comprehensive defect detection, even if it means a slightly increased risk of false positives. This is particularly valuable in quality inspection systems, where undetected defects can lead to critical consequences. Additionally, YOLOv11 has the advantage of high-speed inference, making it appropriate for deployment in real-time industrial environments.

Conversely, InceptionV3 showed significantly lower results, with Accuracy, Precision, and F1-Score at 0.76, and Recall at 0.77. These scores indicate limited

classification capability, suggesting that the model has not effectively learned the necessary features in the context of this specific task. The complexity and depth of InceptionV3 requires a larger and more diverse dataset to fully exploit its potential, something not satisfied by the current dataset. Therefore, InceptionV3 is not currently well-suited for deployment without further refinement or retraining.

In conclusion, VGG16 emerges as the most effective and stable model, striking an excellent balance across all performance metrics. YOLOv11 remains a strong alternative when fast processing and real-time responsiveness are required. Meanwhile, although InceptionV3 is architecturally advanced, it does not deliver optimal results within the current data constraints and task requirements and would require further tuning for continued use.

5.2.2 Comparative ROC Curve Analysis of Classification Models

The Receiver Operating Characteristic (ROC) curve is a crucial tool for evaluating the performance of binary classification models. It illustrates the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) across various classification thresholds. The Area Under the Curve (AUC) is a key quantitative indicator that reflects how effectively a model can distinguish between two classes: the closer the AUC is to 1, the better the model's performance.

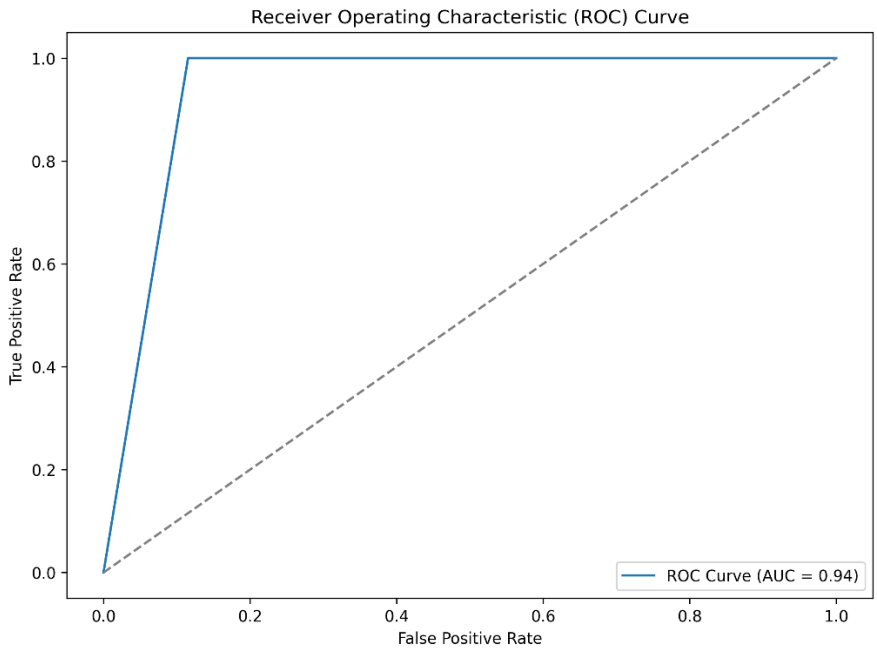


Figure 18. ROC Curve of the YOLOv11 Model

The YOLOv11 model achieves an AUC of 0.94, which is notably strong given that the model was originally designed for object detection rather than classification. The ROC curve sharply bends toward the top-left corner, indicating that the model can

accurately detect most defective products while maintaining a low false positive rate. The smooth and consistent shape of the curve suggests that YOLOv11 performs reliably across various thresholds.

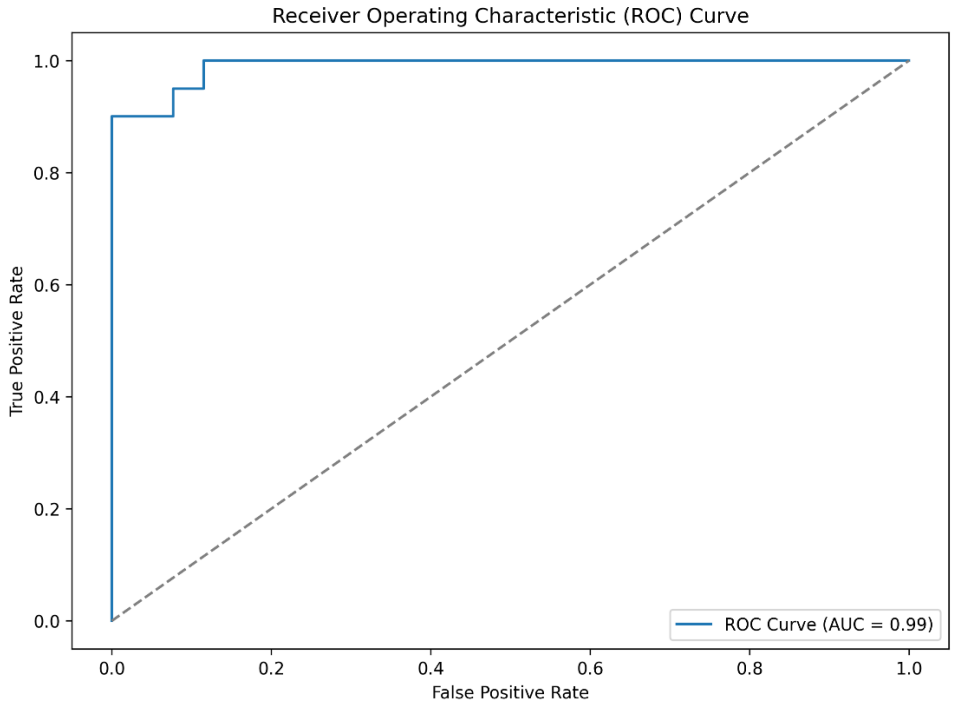


Figure 19. ROC Curve of the VGG16 Model

Figure 18 presents the ROC curve of the VGG16 model, which achieves an AUC of 0.99 -approaching the ideal value. The curve closely follows the y-axis and maintains a TPR near 1.0 across almost all FPR values, indicating near-perfect class discrimination. The steep and smooth curve demonstrates the model's excellent balance between sensitivity and specificity, making it especially well-suited for tasks requiring high precision

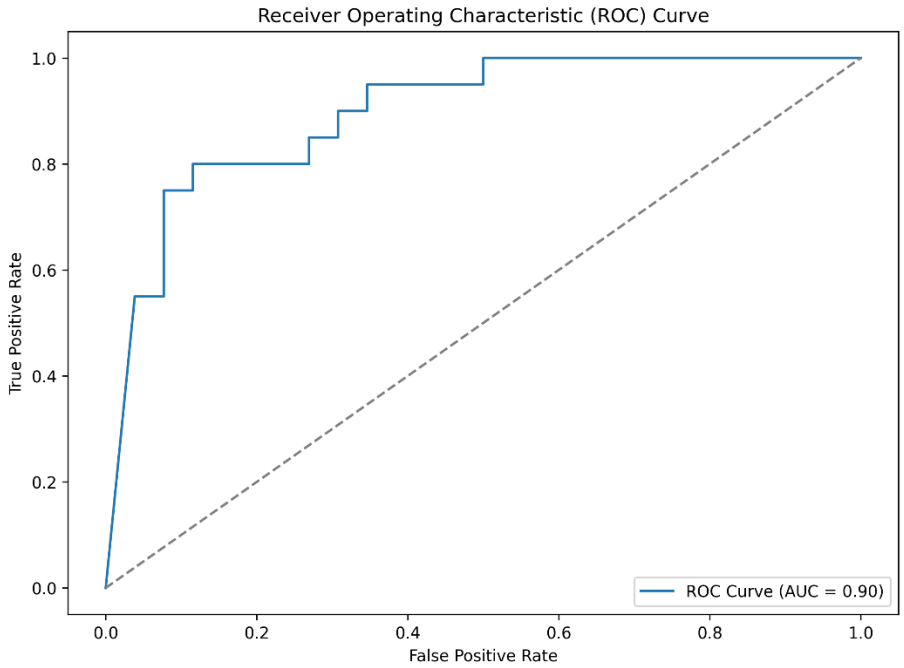


Figure 20. ROC Curve of the InceptionV3

Figure 19 illustrates the ROC curve of InceptionV3, with an AUC of 0.90. While the AUC is still within a strong range, the curve appears more irregular and includes noticeable steps, suggesting inconsistent behavior at different thresholds. Although InceptionV3 maintains acceptable classification performance, its ability to separate classes is less robust than that of YOLOv11 and VGG16.

Based on the ROC curve analysis, it can be concluded that the VGG16 model demonstrates the best performance for the visual defect classification task, in terms of both AUC and curve shape. The YOLOv11 model also performs well and proves to be a versatile choice with consistent classification capabilities. The InceptionV3 model, while still effective, falls slightly behind the other two models in terms of class separation accuracy.

This ROC analysis not only assists in selecting the most appropriate model but also plays a pivotal role in determining the optimal classification threshold, which is critical for enhancing the reliability of automated visual inspection systems in real-world production environments.

5.3 Model Selection and Extension for Defect Classification

After training and evaluating the performance of three models, namely YOLOv11, VGG16, and Inception V3, based on metrics such as accuracy, precision, recall, and F1-score, the results indicated that the VGG16 model outperformed the others in both accuracy and stability. Specifically, VGG16 demonstrated strong classification

capability between the "OK" and "NG" labels, achieving an overall accuracy of 96%, with balanced precision and recall values across both classes.

Based on these results, VGG16 was selected as the primary model for further application in the task of detailed defect classification for products labeled as "NG". In this stage, instead of merely identifying defective products, the system proceeds to recognize specific types of defects. This facilitates root cause analysis and supports improvements in the production process to enhance overall product quality.

5.3.1 Detailed Defect Classification Task Description

After identifying VGG16 as the most suitable model for classifying products into two categories, “OK” and “NG,” this model was further extended to handle the task of detailed defect classification within the “NG” group. The objective of this stage is to accurately identify the specific type of defect present in faulty products, thereby supporting error tracking, root cause analysis, and the proposal of improvements for the production process.

Specifically, five common defect locations were selected for classification: Handle; IO cover; Logo; PR; Silicone button.

This detailed classification task presents several challenges. Notably, there is a data imbalance among the different defect categories, which can cause the model to become biased during training. Furthermore, the visual similarity between certain defect classes increases the complexity of the task, requiring the model to have strong feature extraction capabilities and carefully tuned parameters.

The decision to continue using the VGG16 model at this stage ensures consistency within the overall processing pipeline and leverages its proven image recognition performance from the earlier OK/NG classification phase.

5.3.2 Training Pseudocode

1. Begin
2. Install and Download Dataset from Roboflow
3. Install packages: roboflow, tensorflow, keras
4. Authenticate using Roboflow API key
5. Download dataset version 2 in folder format
6. Set image input size: INPUT_SIZE = (224, 224)
7. Set batch size: BATCH_SIZE = 20
8. Create Data Generators for Training and Validation
9. Preprocessing: normalize pixel values to [0, 1] using rescale = 1/255
10. Use ImageDataGenerator to create training and validation generators

11. Load data from the corresponding directories
12. Load Base Model (VGG16)
13. input_shape = (224, 224, 3)
14. include_top = False
15. weights = 'imagenet'
16. weights = 'imagenet'
17. Add Custom Classification Layers
18. Flatten the output from VGG16
19. Add Dense layer with 512 nodes, ReLU activation

    → $ReLU(x) = max(0, x)$

20. Add Dropout layer with rate = 0.5
21. Output layer: Dense with 6 nodes, Softmax activation

    → $Softmax(x_i) = exp(x_i) / \Sigma exp(x_j)$ (multi-class output)

22. Compile the Model
23. Optimizer: RMSprop with learning_rate = 0.001
24. Loss: categorical crossentropy

    → $L(y, p) = - \Sigma\ y_i \log(p_i)$

25. Metric: Accuracy
26. Define Training Callbacks
27. EarlyStopping: monitor val_loss, patience = 20, restore best weights
28. ReduceLROnPlateau: monitor val_loss, patience = 5, reduce LR with factor = 0.3, min_lr = $1e^{-7}$
29. Custom Callback (LrLogger): record learning rate at the end of each epoch.
30. Train the Model
31. Call model.fit () with:
32. steps_per_epoch = floor (num_train_samples / BATCH_SIZE)
33. epochs = 100

    a. callbacks: early stopping, LR scheduler, learning rate logger

34. Evaluate ROC Curve
35. Predict class probabilities (y_pred_probs) on validation set
36. Extract ground truth: y_true
37. Calculate ROC Curve: fpr, tpr
38. Computer AUC score: ∫ ROC

39. Plot Learning Rate and Training History
40. Plot learning rate over epochs
41. Plot training and validation loss/accuracy over epochs
42. Save the Plots
43. Final Model Evaluation
44. Use model. evaluate () on the validation set
45. Predict class labels using argmax (softmax output)
46. Generate and display:
47. Confusion Matrix
48. Classification Report
49. Convert Model to TensorFlow Lite
50. Use TFLiteConverter.from_keras_model ()
51. Apply optimization: tf.lite.Optimize.DEFAULT
52. Export and save the model as VGG16.tflite
53. End

5.3.3 Defect Classification Evaluation

Table 7. Classification Report

	Precision	**Recall**	**F1-Score**	**Time**
Good	1,00	0,92	0,96	
Handle	0,8	0,8	0,8	
IO Cover	1,00	0,67	0,8	
Logo	0,83	1,00	0,91	11s
PR	0,86	1,00	0,92	
Silicone buttons	1,00	1,00	1,00	
Macro avg	0,91	0,9	0,9	

The evaluation results indicate that the model achieved relatively high precision and recall across most classes. Specifically, the "Good" class reached a precision of 1.00 and a recall of 0.92, suggesting that the model can accurately identify non-defective samples with minimal false positives.

For defect classes such as "Logo", "PR", and "Silicone buttons", the model achieved a perfect recall of 1.00, meaning it successfully detected all instances of those defect types. However, the "IO Cover" class showed a recall of only 0.67, indicating that some samples in this class were misclassified as other types.

Most classes had precision scores ranging from 0.80 to 1.00. Notably, the "Handle" class had the lowest precision at 0.80, possibly due to class imbalance or overlapping visual features with other defects.

The overall macro F1-score was 0.90, which reflects the model's balanced performance across all categories. Despite this, slight discrepancies between precision and recall in some classes highlight the need for further refinement in data collection and model training to ensure more consistent classification results.

In the process of applying the Convolutional Neural Network (CNN) for defect detection, processing time is an important factor that affects the system's ability to work in real production lines. Each input image takes about 7 seconds for the model to analyze and return the classification result.

The system uses a rotating table to capture different sides of the product. For each product, three sets of images are taken to cover six sides. These images are processed at the same time to improve efficiency. To capture all three views, the turntable needs to rotate twice, and each rotation takes around 2 seconds, so the total rotating time is 4 seconds.

As a result, the total time to complete the inspection of one product is around 11 seconds (including image processing and rotation). This is a reasonable processing time for production lines with medium speed and shows that the system can be improved and applied in real manufacturing environments.

5.4 Automated defect inspection system

To improve product consistency and reduce human error in quality inspection, we developed an automatic visual inspection system specifically for Bluetooth speakers. Instead of relying on manual checking, the system uses a conveyor-integrated rotary table, and three industrial cameras positioned to capture key areas of the speaker during its movement through the inspection zone. Each camera is responsible for different perspectives, ensuring a complete and efficient evaluation of the speaker's external condition.

A top-down camera is mounted vertically above the conveyor, focusing on inspecting the silicone buttons and the speaker's handle area. These components are essential user-interaction zones, requiring detection of defects such as dirt, surface scratches, or assembly misalignment.

Two side-view cameras are fixed on both sides of the conveyor using external support arms to avoid interfering with the moving line. Camera 1 is configured to inspect the left Passive Radiator (PR) surface and the logo surface, identifying defects such as scratches, stains, or logo misalignment. Camera 2 is responsible for capturing images of the right PR surface and the IO port area. The IO region is particularly checked for cosmetic issues including scratches, dust, or other contamination that may affect usability or aesthetics.

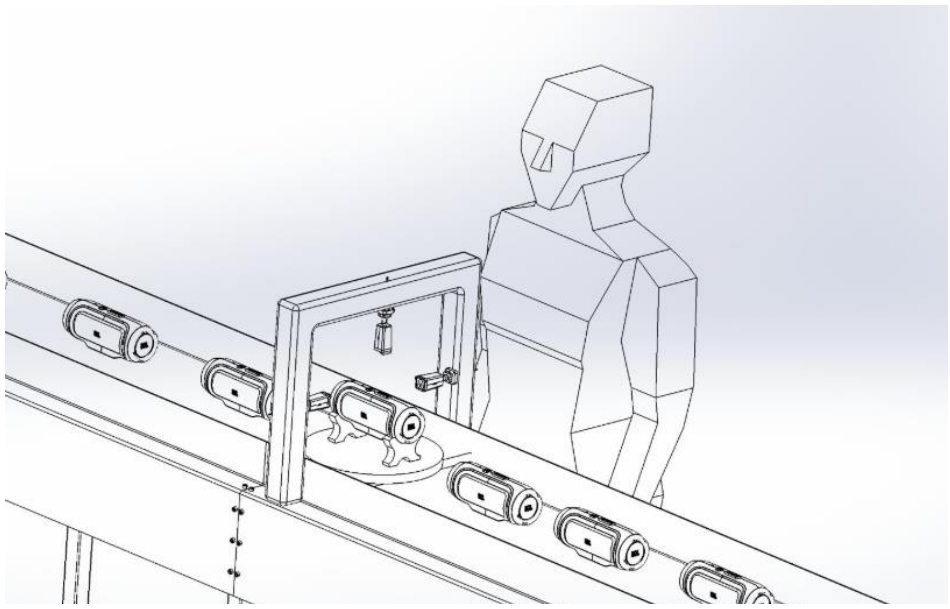


Figure 21. Visual Inspection System on Production Conveyor

After completing the acoustic inspection stage, the Bluetooth speaker product is transferred to the visual inspection station. At this stage, an operator places the product on a rotating platform equipped with a camera system designed to capture images of the product from multiple angles for the purpose of visual quality assessment.

The system employs a total of three cameras: one mounted above the platform to capture the top view of the product, and two side-mounted cameras to capture lateral views. Thanks to the platform's rotational mechanism, the side cameras can sequentially acquire images of all surrounding surfaces of the product. The rotation process is carried out in 90-degree increments to ensure comprehensive and consistent image coverage. Once the image acquisition process is complete, the platform automatically returns to its initial position, ready for the next inspection cycle.

The captured images are transmitted to a central processing unit, where a deep learning model based on Convolutional Neural Networks (CNN) analyzes and evaluates

the external appearance of the product. The model performs a classification task, sorting the product into one of the following categories:

- **OK:** No visual defects are detected, and the product meets the quality standards required to proceed to the next production stage.
- **NG:** Visual defects are detected. The system displays the classification result on the screen, indicating the defective region such as the handle, IO cover, logo area, or silicone buttons.

Based on the output displayed by the system, the operator takes appropriate action. For products classified as OK, the operator returns the item to the output conveyor for the next production step. For those identified as NG, the product is placed into the designated defect bin to await repair or further handling according to factory procedures.

CHAPTER 6: CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

This capstone project has successfully developed an automated visual inspection system for Bluetooth speaker manufacturing using Convolutional Neural Networks (CNN). The system was designed to replace traditional manual inspection methods, which are often time-consuming, inconsistent, and highly dependent on human judgment.

The system integrates industrial cameras and a motorized turntable to automatically capture images of each product from multiple angles. These images are then analyzed by a two-stage deep learning model. The first stage identifies whether the product is defective or not, while the second stage determines the location of the defect on the product surface. This structured approach helps ensure the reliability and consistency of the inspection process.

In terms of performance, the proposed solution reduced the average inspection time per product from approximately 32 seconds (manual inspection) to around 11 seconds, including both image processing and rotation time. Additionally, the number of operators needed for the visual inspection process was reduced from two to one, leading to improved labor efficiency and reduced production costs.

Moreover, the system demonstrates the ability to maintain stable inspection quality across multiple test cases, supporting its potential for long-term use and scalability in real manufacturing environments.

In conclusion, the application of CNN-based models in automated visual inspection not only improves inspection speed and reliability but also supports the broader trend of integrating artificial intelligence into manufacturing. This project provides a practical foundation for further development and optimization of intelligent quality control systems in the electronics industry and beyond.

6.2 Recommendations

Based on the results achieved throughout the project, several recommendations are proposed to enhance the effectiveness of the automatic defect detection system:

- Expand and diversify the training dataset: Increasing the volume and variety of defect images will allow the model to learn more features, thereby improving classification performance, especially for defect types with lower occurrence rates.
- Improve classification accuracy for specific defect classes: Some defect classes, such as “IO Cover” or “Handle,” still exhibit relatively low precision. Enhancing data quality and labeling accuracy for these classes is necessary to improve the overall classification results.

- Currently, the system takes about 11 seconds to process one product. To better meet the speed requirements of actual production lines, it is necessary to explore model optimization methods to reduce processing time while maintaining accuracy.

REFERENCES

[1] J. Van der Tempel and G. Nils, "Deep learning for automated visual inspection of surface defects: A survey," 2023. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0166361523000611. [Accessed 12 June 2025].

[2] Y. Z. C. D. X. Liu, "A survey of real-time surface defect inspection methods based on deep learning," 2024. [Online]. Available: https://link.springer.com/article/10.1007/s10462-023-10475-7. [Accessed 12 June 2025].

[3] J. F. T. Z. R. L. P. C. J. M. Y. S. Zirou Jiang, "Defect R-CNN: A Novel High-Precision Method for CT Image Defect Detection," 26 April 2025. [Online]. Available: https://surl.lu/pkjrhc.

[4] N. Y. Jingyao Wang, "SSD-Faster Net: A Hybrid Network for Industrial Defect Inspection," 3 July 2022. [Online]. Available: https://arxiv.org/abs/2207.00589. [Accessed 23 May 2025].

[5] A. L. F. G. P. G. D. P. A. Calabrese M., "Application of Mask R-CNN and YOLOv8 Algorithms for Defect Detection in Printed Circuit Board Manufacturing," 24 March 2025. [Online]. Available: https://surl.li/ewvgsj. [Accessed 23 May 2025].

[6] Z. Y. W. Z. Z. X. Z. Y. Li X., "A Compact Convolutional Neural Network for Surface Defect," 2 April 2020. [Online]. Available: https://surl.li/adbpvr. [Accessed 23 May 2025].

[7] G.-S. M. J. d. l. E. A. García-Pérez A., "Automated Defect Recognition of Castings Defects Using Neural Networks," 6 September 2022. [Online]. Available: https://arxiv.org/abs/2209.02279. [Accessed 23 May 2025].

[8] X. X. Y. L. Y. B. Wang S., "Automatic Detection and Classification of Steel Surface Defect Using Deep Convolutional Neural Networks," 5 March 2021. [Online]. Available: https://surli.cc/stvhqn. [Accessed 23 May 2025].

[9] Z. Q. Z. K. L. H. Z. Y. C. W. Deng M., "A Novel Defect Inspection System Using Convolutional Neural Network for MEMS Pressure Sensors," 30 September 2022. [Online]. Available: https://surl.li/ryosvt. [Accessed 23 May 2025].

[10] W. contributors, "Confusion matrix," Wikipedia, 5 December 2024. [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix. [Accessed 12 June 2025].

[11] R. K. a. M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," arXiv, Ithaca, NY, 2024.

[12] S. Avasthi, "YOLOv11: Model Building and Training," 5 January 2025. [Online]. Available: https://surl.li/vhbkdy. [Accessed 25 May 2025].

[13] A. Jha, "VGG-16 CNN Model for Classification and Detection," TechCraft, 16 October 2023. [Online]. Available: https://surl.li/yvbsmi. [Accessed 25 May 2025].

[14] A. Das, "Inception V3 model architecture," OpenGenus, 30 March 2019. [Online]. Available: https://surl.lu/jxvxtw. [Accessed 25 May 2025].

[15] R. K. a. M. Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," arXiv, Ithaca, New York, 2024.

[16] W. contributors, "Confusion matrix," 2024. [Online]. Available: https://en.wikipedia.org/wiki/Confusion_matrix?utm_source. [Accessed 15 June 2025].