



ALGERIAN DEMOCRATIC AND POPULAR REPUBLIC  
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC  
RESEARCH  
KASDI MERBAH UNIVERSITY OUARGLA  
FACULTY OF NEW TECHNOLOGIES OF INFORMATION  
AND COMMUNICATION  
DEPARTMENT OF ELECTRONICS AND  
COMMUNICATIONS



**ACADEMIC MASTER thesis**  
**Domain: Science and Technology**  
**Field: Electronics**  
**Speciality: Electronics of Embedded Systems**  
by  
**Oussama GHERIER**  
**Oualid BENHANIA**  
**Boukhari KHADRAOUI**

## ***THEME***

---

# ***Using Convolutional Neural Networks to detect defects in manufacturing***

---

Submit Date : June 2024

Before the jury :

**Dr.Azzedine HAMZA**  
**Dr.Abdelmadjid YUCEFA**  
**Dr.Azeddine BENLAMOUDI**

**President**  
**Supervisor**  
**Examiner**

**UKM Ouargla**  
**UKM Ouargla**  
**UKM Ouargla**

University year : 2023/2024

## ***Acknowledgment***

*First we would thank the one above all of us, the omnipresent GOD, for answering our prayers and for giving us the strength to plod thank you so much GOD. We express our sincere gratitude to our supervisor, Dr. Abdelmadjid Youcefa, for the continuous support, sage advice, insightful criticisms, and encouragement. He always motivates us to solve our research problems. His guidance helped us in all the time of research and writing of our thesis. And special thanks for his confidence in us. Likewise, we extend our respectful thanks to the jury members for accepting to review our work we are honored to have you and we thank you before for your remarks. We sincerely thank all our teachers who taught us through our course. Also for giving us the opportunity to show our capacities, thanks to all. There are no words to express our gratitude and thanks to our Beloved parents, for supporting us in our daily lives, for guiding us always, and for their prosperity and love. A special thanks to our family and friends for always standing by us. Their love has been the major spiritual support in our lives. As a final word, we would like to thank each and every individual who has been a source of support and encouragement and helped us to achieve our goal and complete our dissertation work successfully.*

## ***Dedication***

*Thanks be to God, Lord of all worlds. Praise be to God in any case, after an academic process that brought with it many difficulties, hardship, and fatigue.*

### ***Oussama:***

*I am honored to dedicate this graduation to my parents, the two people who provided me with the tools and values necessary to reach where I am today, to my brothers who supported me, to all my colleagues who I met along the way, and to everyone who helped me in my career.*

### ***Boukhari:***

*In the name of God, the Most Gracious, the Most Merciful To my dear parents, without whose constant support and unlimited encouragement I would not have been able to achieve this achievement. To my esteemed professor, who spared no effort in guiding me and providing me with knowledge. To my colleagues and friends, who have always supported me in my career. With all gratitude and appreciation, I dedicate this work to you as an expression of my thanks and pride.*

### ***Oualid:***

*To those who believed in me and supported me on my learning journey, to those who lit my path and supported me at every step, to my family, friends, and teachers, I dedicate to you the fruit of my effort and the joy of my graduation. Thank you for supporting me in the most difficult moments and motivating me to achieve my dreams. with all gratitude and appreciation.*

## Abstract

The manufacturing industry is undergoing a technical revolution thanks to advanced developments in deep learning (DL) techniques. This study makes an important contribution to the field of automatic defect detection by applying CNNs, focusing on the YOLOv8 and YOLOv9 models.

These two models were designed and trained on a custom dataset of PCB images with the aim of detecting defects with high accuracy and efficiency. The results showed significant superiority of both models over traditional methods in the defect detection task. Moreover, the use of “transfer learning” technology has proven to be very effective in accelerating the training process and significantly improving the performance of the two models while relying on a smaller amount of training data.

Through the experiments that we conducted, we recorded a difference in the results obtained for the two models according to the standards. In terms of accuracy, they achieve approximately the same percentage, but in terms of the time taken for training and classification, the YOLOv8 model is better than YOLOv9. As for recall, the YOLOv9 model is better than YOLOv8. In each case, the criterion must be determined. The appropriate trial will determine which of the two models is most effective for detecting defects and practical assistance in manufacturing.

CNNs, especially YOLOv8 and YOLOv9, have been able to achieve high accuracy in detecting PCB defects.

The use of transfer learning also contributed to accelerating the training process and improving the performance of the two models.

### **Keywords:**

Deep Learning, CNN, Transfer Learning, YOLOv8, YOLOv9, Defect Detection in Manufacturing, PCB

## ملخص

يشهد مجال التصنيع ثورة تقنية بفضل التطورات المتقدمة في تقنيات التعلم العميق (DL). تُقدم هذه الدراسة مساهمة هامة في مجال الكشف التلقائي عن العيوب من خلال تطبيق شبكات CNN، مع التركيز على نمودجي YOLOv8 و YOLOv9. تم تصميم وتدريب هذين النمودجين على مجموعة بيانات مُخصصة من صور ثنائي الفينيل متعدد الكلور (PCB) بهدف الكشف عن العيوب بدقة وكفاءة عالية. أظهرت النتائج المُتوصَّل إليها تفوقًا ملحوظًا لكلا النمودجين على الأساليب التقليدية في مهمة الكشف عن العيوب. علاوة على ذلك، أثبت استخدام تقنية "التعلم بالنقل" فعالية كبيرة في تسريع عملية التدريب وتحسين أداء النمودجين بشكل ملحوظ مع الاعتماد على كمية بيانات تدريبية أقل. من خلال التجارب التي قمنا بها سجلنا تباين في النتائج المحصل عليها للنمودجين حسب المعايير فمن ناحية الدقة يحققان نفس النسبة تقريبًا أمّا من ناحية المدة المستغرقة للتدريب و التصنيف فنمودج YOLOv8 أحسن من YOLOv9، أمّا من ناحية الإستدعاء فنمودج YOLOv9 أحسن من YOLOv8، على كلّ تحديد المعيار المناسب للتجربة هو الذي يحدد أيّ النمودجين هو الأفضل فعاليةً للكشف عن العيوب والمساعدة العملية في مجال التصنيع. تمكنت شبكات CNN، وخاصةً YOLOv8 و YOLOv9، من تحقيق دقة عالية في الكشف عن عيوب ثنائي الفينيل متعدد الكلور (PCB). كما ساهم استخدام التعلم بالنقل "Transfer Learning" في تسريع عملية التدريب وتحسين أداء النمودجين.

الكلمات المفتاحية: التعلم العميق، CNN، Transfer Learning، YOLOv8، YOLOv9، الكشف عن العيوب في التصنيع، PCB

## Resumé

L'industrie manufacturière connaît une révolution technique grâce aux développements avancés des techniques d'apprentissage profond (DL). Cette étude apporte une contribution importante au domaine de la détection automatique de défauts en appliquant des CNN, en se concentrant sur les modèles YOLOv8 et YOLOv9. Ces deux modèles ont été conçus et entraînés sur un ensemble de données personnalisé d'images de PCB dans le but de détecter les défauts avec une grande précision et efficacité. Les résultats ont montré une supériorité significative des deux modèles par rapport aux méthodes traditionnelles dans la tâche de détection des défauts. De plus, l'utilisation de la technologie « d'apprentissage par transfert » s'est avérée très efficace pour accélérer le processus de formation et améliorer considérablement les performances des deux modèles tout en s'appuyant sur une plus petite quantité de données de formation. A travers les expériences que nous avons menées, nous avons enregistré une différence dans les résultats obtenus pour les deux modèles selon les normes. En termes de précision, ils atteignent à peu près le même pourcentage, mais en termes de temps nécessaire à la formation et à la classification, le modèle YOLOv8 est meilleur que YOLOv9, et en termes de rappel, le modèle YOLOv9 est meilleur que YOLOv8. Dans chaque cas, le critère doit être déterminé. L'essai approprié déterminera lequel des deux modèles est le plus efficace pour détecter les défauts et fournir une assistance pratique à la fabrication. Les CNN, en particulier YOLOv8 et YOLOv9, ont réussi à atteindre une grande précision dans la détection des défauts des PCB. Le recours à l'apprentissage par transfert a également contribué à accélérer le processus de formation et à améliorer les performances des deux modèles.

Mots clés : Deep Learning, CNN, Transfer Learning, YOLOv8, YOLOv9, Détection de défauts dans la fabrication, PCB.

# Contents

<b>List of Figures</b>	<b>VII</b>
<b>Chapter 1: General Introduction</b>	<b>2</b>
1.1 Introduction . . . . .	2
1.1.1 Background and Motivation . . . . .	2
1.1.2 Problem . . . . .	2
1.1.3 Solution . . . . .	3
1.2 Thesis structure: . . . . .	4
<b>Chapter 2: Literature review</b>	<b>6</b>
2.1 Introduction . . . . .	6
2.2 Related Work . . . . .	6
2.3 Conclusion . . . . .	9
<b>Chapter 3: Methodology</b>	<b>11</b>
3.1 Introduction . . . . .	11
3.2 Machine learning . . . . .	12
3.2.1 Supervised learning: . . . . .	12
3.2.2 Unsupervised Learning: . . . . .	14
3.3 Neural networks . . . . .	16
3.3.1 Loss Function: . . . . .	16
3.3.2 Backpropagation . . . . .	17
3.4 Deep learning . . . . .	17
3.5 convolution neural networks (CNNs) . . . . .	18
3.5.1 Convolutional Layer . . . . .	19

3.5.2	Poling layer . . . . .	22
3.5.3	Fully Connected Layers . . . . .	23
3.6	Transfer Learning . . . . .	23
3.7	Some of the concepts and techniques used: . . . . .	24
3.7.1	One-hot encoding . . . . .	24
3.7.2	Softmax activation function . . . . .	25
3.7.3	Dropout . . . . .	26
3.7.4	Batch normalization . . . . .	26
3.7.5	Gradient descent . . . . .	26
3.7.6	Stochastic Gradient Descent(SGD) . . . . .	27
3.7.7	Flatten layers . . . . .	28
3.8	Proposed Method: . . . . .	28
3.8.1	YOLOv8: . . . . .	29
3.8.2	YOLOv9: . . . . .	30
3.9	Conclusion: . . . . .	33
<b>Chapter 4: Results and Descussion</b>		<b>35</b>
4.1	Introduction . . . . .	35
4.2	Data collection . . . . .	35
4.3	Performance Metrics: . . . . .	36
4.4	Results : . . . . .	37
4.5	Descussion: . . . . .	41
4.6	Conclusion: . . . . .	42
<b>Chapter 5: General Conclusion:</b>		<b>44</b>
5.1	Conclusion: . . . . .	44
<b>References</b>		<b>46</b>



## List of Figures

3.1	The relationship between CNN and deep learning and machine learning . . . . .	11
3.2	Type of Machine learning . . . . .	12
3.3	Supervised learning Workflow . . . . .	13
3.4	Decision Tree . . . . .	13
3.5	Support Vector Machine . . . . .	14
3.6	Unsupervised Learning . . . . .	14
3.7	Principal Component Analysis . . . . .	15
3.8	K-Means Clustering . . . . .	15
3.9	A 3-layer neural network with three inputs . . . . .	16
3.10	Machine learning VS deep learning . . . . .	17
3.11	CNN-architecture . . . . .	18
3.12	3D data input . . . . .	19
3.13	feature map . . . . .	20
3.14	Example of Stride . . . . .	21
3.15	padding . . . . .	21
3.16	RELU . . . . .	22
3.17	polling . . . . .	23
3.18	Fully Connected Layers . . . . .	23
3.19	The architecture of transfer learning model. . . . .	24
3.20	data set . . . . .	25
3.21	One-hot encoding . . . . .	25
3.22	softmax . . . . .	25

---

3.23	droupout . . . . .	26
3.24	Gradient descent . . . . .	27
3.25	Flowchart of the proposed method . . . . .	28
3.26	YOLOv8 architecture diagram. . . . .	30
3.27	Visualization results of random initial weight output feature maps for different network architectures[1] . . . . .	30
3.28	Programmable Gradient Information (PGI) Architecture[1] . . . . .	32
3.29	Generalized Efficient Layer Aggregation Network (GELAN)[1] . . . . .	33
4.1	Input images with dissimilar defects on PCB: missing hole;spurious copper;open circuit;mouse bite;short;spur. . . . .	35
4.2	Splitting the dataset. . . . .	36
4.3	F1 Score Curve . . . . .	37
4.4	Precision-Recall Curve . . . . .	37
4.5	Precision Curve . . . . .	38
4.6	Recall Curve . . . . .	38
4.7	Confusion Matrix . . . . .	38
4.8	Validation Batch Labels YOLOv8 . . . . .	39
4.9	Validation Batch Labels YOLOv9 . . . . .	39
4.10	Validation Batch Predictions YOLOv8 . . . . .	40
4.11	Validation Batch Predictions YOLOv9 . . . . .	40
4.12	Results . . . . .	41

**CHAPTER 1**

---

**General Introduction**

---

# Chapter 1

## General Introduction

### 1.1 Introduction

#### 1.1.1 Background and Motivation

Within a manufacturing plant, manufacturing defect detection refers to the systems and processes in place to prevent defective items from reaching customers.

In most cases, defect detection is accomplished through a visual defect inspection—in-line or end-of-line. Quality-assurance staff are simply questioning if the product coming off the production line meets all criteria.

Quality control is critical in any manufacturing organization. If you work in manufacturing, you already know that the lower your defect rate, the better. Furthermore, identifying difficulties early in the process allows you to save more time and money.

Most manufacturing organizations are built around a dedication to creating high-quality products. It's easy to understand why. You have a reputation to preserve, yet in some businesses, poor quality standards can cause harm or even death.

Anything you can do to lower the price will boost your margins. One of the most effective methods available today is to improve the effectiveness of fault identification in production.

Defect detection can save money by reducing wastage of raw materials, protecting your reputation, and avoiding fines or recalls. Manufacturers are searching for a competitive advantage wherever they can find one, and smarter manufacturing defect detection is simpler than you may believe.

#### 1.1.2 Problem

Addressing manufacturing defects involves numerous challenges. Acquiring accurate and comprehensive data regarding defects can be difficult, especially for subtle or rare issues. Classifying defects accurately can be complicated by the diversity of defects and the similarity of some defects to normal variations. Moreover, detecting defects in real-time during production requires rapid and reliable analysis methods. Variability in manufacturing processes, such as

changes in materials or environmental conditions, can also pose challenges to defect detection. Additionally, integrating defect detection systems into existing production lines and quality control processes may require significant adjustments. Balancing the costs of implementing defect detection systems with the potential savings from reduced defects is another crucial consideration for manufacturers.

### 1.1.3 Solution

Inaccurate product categorization is an important aspect of quality control in manufacturing. This job is currently being addressed using a variety of ways. Defective product categorization systems must meet stringent criteria, including real-time accuracy and robust performance in noisy environments such as factories. As machine learning and vision systems improve, feature-based defect classification algorithms such as artificial neural networks (ANN), Bayesian network classifiers, and support vector machines (SVM) are being studied and applied to identify defective products [2, 3]. In real-world factories, feature-based classification systems may be affected by noise, including illumination variations and shadows in pictures. Furthermore, some organizations produce a wide range of things across multiple product lines, making feature-based approaches ineffective. Deep learning techniques have made significant advances in computer vision applications such as object recognition, image categorization, and object identification. These achievements hold substantial promise for usage in production [4, 5]. A multitask CNN is used to identify wire defect regions and categorize damaged products [6, 7]. CNNs have been used for quality inspection of a variety of products, including PCBs [8, 9], metal surfaces [10], bottled wine, casting products, semiconductor fabrication, LED cup apertures, mobile phone screens, display panel cover glass, bearings, optical film, and leather defect.

This project aims to classify defects using a deep learning model (CNN). Convolutional neural networks (CNNs) provide an appealing approach for directly addressing production flaws in today's industrial scene. With their ability to understand complicated patterns and abnormalities from vast data sets, CNNs provide a cutting-edge solution to anomaly identification. CNN models can detect minor visual cues indicating faults with surprising accuracy after training on large sets of photos representing defective and defect-free products. Furthermore, CNNs can interpret data from a variety of sensors and production processes, allowing for real-time monitoring and early detection of any faults before they worsen. Integrating CNN-powered defect detection systems into production workflows automates quality assurance, reduces the risk of defective products, and raises product quality standards to new heights, increasing consumer confidence and loyalty.

## 1.2 Thesis structure:

The work is listed as follows: The second chapter presents the experiments conducted by researchers to detect defects using different techniques. . Chapter 3 introduces machine learning, deep learning, and transfer Learning, CNN architecture and YOLO architecture to improve model training Some other techniques used to obtain a high learning rate are also explained. Chapter Four presents the results obtained by training the dataset for PCB defects on YOLOv8 and YOLOv9, and finally, Chapter Five presents a general conclusion.

**CHAPTER 2**

---

**Literature review**

---

# Chapter 2

## Literature review

### 2.1 Introduction

This work is intended to study the approach of defective classification done on smart factories to enhance their level through various deep learning methods. Those are discussed by several researchers, and a few are discussed below.

### 2.2 Related Work

Alexey N. Beskopylny et al.[11] proposed an intelligent algorithm based on convolutional neural network (CNN) to detect and classify defects in facing brick specimens. The algorithm uses YOLOv5s CNN, which is known for its accuracy and speed in object detection. It involves creating a database of images of facing brick specimens, implementing the algorithm, optimizing it, debugging it, and testing it using CNN. The results showed that the algorithm is able to detect and classify various defects, such as foreign inclusions, broken corners, cracks, and color inconsistencies. The model achieves high accuracy in detecting defects with mAP values of 87 and 72. It emphasizes the practical importance of the algorithm in improving the efficiency and accuracy of visual inspection in the production and construction stages of facing brick materials. In general, it contributes to the development of artificial intelligence methods in the field of construction and the development of intelligent systems for detecting defects.

Compare Yi-Cheng Huang, Kuo-Chun Hung, and Jun-Chang Lin in detecting defects on cylindrical metal surfaces.[12]Compare three convolutional neural networks (VGG-16, ResNet-50, and MobileNet v1) in detecting defects on metal surfaces using Vector learning (TL) explores the phenomenon of apparent convergence in prediction accuracy followed by divergence in validation. Accuracy when using TL. They also develop an Automated Machine Learning (AutoML) model using random search with the underlying layers of the three machine learning models. In addition, they use AutoKeras to implement AutoML and compare its performance with TL and the designed AutoML models. It highlights the strengths and weaknesses of each



model. VGG-16 has the highest accuracy but requires a large number of parameters and a long training time. ResNet-50 shows low verification accuracy and fluctuating prediction accuracy. MobileNet v1 has a low level of training and validation accuracy. AutoKeras achieves the highest levels of accuracy but requires a long training time. An AutoML model built using the MRV model architecture achieves a final accuracy of 95.5. Overall, the researchers provide a comprehensive analysis of the different models and their performance in detecting defects on metal surfaces. They demonstrated the advantages of using TL and AutoML in improving defect detection and reducing training costs.

Discusses a wire bonding defect detection algorithm called YOLO-CSS for X-ray images[13]. The algorithm is based on an improved version of the YOLOv8 network and incorporates a feature extraction module called C2SF, which effectively captures semantic features from different gradient information. The network also includes a self-adaptive weighted multi-scale feature fusion module called SMA, which assigns different weights to feature maps of different scales based on their contribution to the detection results. Additionally, the network utilizes skip connections to maintain the integrity of feature information. The algorithm achieves high detection accuracy and recall rates, outperforming other mainstream algorithms. Future research directions include exploring deep-learning-based algorithms for few-shot learning and improving image preprocessing techniques to enhance defect localization and recognition.

Comprehensively surveyed deep learning methods in anomaly detection Jing Yang, et al[14]. They classify defects in different products and review prevailing techniques and deep learning methods for defect detection. They summarize and analyze the application of ultrasonic testing, filtering, deep learning, machine vision, and other techniques in defect detection. They also examined the functions and characteristics of existing defect detection equipment. The strengths of deep learning methods in anomaly detection include their ability to learn abstract and basic features, extract high-level semantic knowledge, and provide accurate and real-time detection. In addition, these methods allow the detection of small objects and complex backgrounds. However, there are challenges and weaknesses that need to be addressed, such as the need for more training data, the complexity of network structures, and the possibility of over-fitting. Overall, their research provides a comprehensive overview of the state of research in defect detection technology in complex industrial processes. It highlights the potential of deep learning methods to improve product quality and reduce production costs. However, they note that more research is needed in areas such as online defect detection, 3D defect detection, and multilayer shape detection. They also suggest incorporating different non-destructive defect detection methods and developing intelligent defect detection equipment.

Huang, W.C. The authors, Cheung and Yi Su, propose a method to detect defects in textile manufacturing using deep learning techniques. Their research focuses on using deep learning algorithms, specifically the UNet++ network architecture, to accurately segment defects in histological images. They first describe the dataset used, which includes a self-made dataset of

textile images with labeled defects, as well as two public datasets, the knitting dataset and the nanofiber dataset. They then explain the methodology used, which includes training neural network models on labeled datasets, performing defect classification and segmentation using sliding window and UNet++ techniques, and evaluating the performance of the models. Strengths of the proposed method include high defect detection accuracy rates, reaching 99.55 on the knitting dataset. The UNet++ network architecture is able to accurately predict fault topography, but comes with a slower prediction time compared to other models. Experimental results showed promising results, especially with regard to recall rates for detecting defects. However, there are also some weaknesses in the proposed method. Accuracy rates are flawless images are relatively low, leading to a higher number of false positives. In addition, the dataset size for some tissue types is limited, which may affect the generalizability of the models. Overall, the research provides valuable insights into the use of deep learning techniques for automated defect detection in textile manufacturing. The proposed method shows promising results in improving screening accuracy and reducing costs. However, more research is needed to overcome the limitations and improve the performance of the models on defect-free images.

In a study by Chen et al., [15], they used deep learning techniques to enhance defect detection in an additive manufacturing process. The research used Efficient Net and Mask R-CNN models for image classification and defect segmentation, respectively. The strengths of the study include developing a comprehensive system for detecting defects without the need for manual adjustments and achieving high accuracy rates. However, potential weaknesses may include the need for further validation in real-world manufacturing settings and exploration of additional deep learning models to improve performance. Overall, the research presents a promising approach for enhancing quality control in additive manufacturing processes, with implications for increasing efficiency and accuracy in defect detection.

The study conducted by Uijlings, J.R.R. et al. introduces a novel approach for defect detection in printed circuit boards (PCBs) using You-Only-Look-Once Convolutional Neural Networks (YOLO CNNs). [8]The method evaluates detection performance based on various criteria such as accuracy, misclassification rate, true positive rate, false positive rate, true negative rate, precision, and prevalence. The strengths of the proposed approach lie in achieving a high PCB defect detection accuracy of 98.79, surpassing the performance of other algorithms that involve complex feature extraction. However, potential weaknesses may include the need for a large and diverse dataset for robust training and the computational resources required for deep learning models. Overall, the study highlights the effectiveness of deep learning techniques, specifically YOLO CNNs, in enhancing the quality inspection process of PCBs, but further research is warranted to address scalability and generalizability concerns.

In the study conducted by Zhifen Zhang et al., the research focused on weld image deep learning-based on-line defects detection using Convolutional neural networks for Al alloy in robotic arc welding. [16]The author employed a CNN classification model with 11 layers to

identify weld penetration defects in real-time for Al alloy in pulsed GTAW. The method involved data augmentation, noise addition, and image rotation to enhance the CNN dataset, improving classification accuracy by approximately 3.88. Additionally, the deep learning image features were visualized and analyzed using the Non-zero Pixel (NOP) method, providing a clear explanation of their physical meaning. Strengths of the study include the innovative approach of utilizing deep learning techniques for real-time weld defect detection, the incorporation of data augmentation to enhance model robustness, and the visualization of CNN features using the NOP method. Furthermore, capturing weld images from different welding conditions contributed to improving the generalization ability of the CNN model. However, some weaknesses can be identified, such as the challenges faced in obtaining clear vision of the welding pool under intense arc light exposure and the dynamic changes in arc light intensity during welding, which may affect the model's performance. Additionally, the study could benefit from further exploration of different welding conditions and defect types to enhance the model's versatility. In general, the research by Zhifen Zhang et al. presents a valuable contribution to the field of robotic arc welding by demonstrating the potential of CNN-based defect detection in real-time scenarios. By addressing the challenges and complexities associated with weld defect detection, the study opens up opportunities for further advancements in automated manufacturing processes.

## 2.3 Conclusion

This section goes into several approaches for defect detection in various types of factories, emphasizing the findings of numerous studies on machine learning techniques and Convolutional Neural Networks (CNNs) for product defect detection. Multiple studies have shown that using CNNs for image processing-based defect detection improves accuracy dramatically. As a result, focusing on CNNs has been extremely useful in accurately diagnosing damaged products.

**CHAPTER 3**

---

**Methodology**

---

# Chapter 3

## Methodology

### 3.1 Introduction

Defect detection in manufacturing using Convolutional Neural Networks (CNN) is a modern and effective technology used to improve the quality of manufactured products. This technology relies on advanced electronic components such as cameras, image processing units, and powerful computers, in addition to intelligent software to develop and train neural networks. The process of detecting defects is to take high-resolution images of manufactured products, then process them using image processing units to improve their quality and convert them into usable data by the neural network. Next, the data is fed into the neural network, which has been pre-trained to recognize different types of defects in manufactured products. Thanks to the ability of neural networks to analyze images with high accuracy, detected defects are classified according to their type and severity. The classification results are then output to the user, who can use them to take action to improve production quality and prevent defects from recurring in the future. In this chapter we introduce CNNs, where CNN falls under deep learning, as shown in the figure:

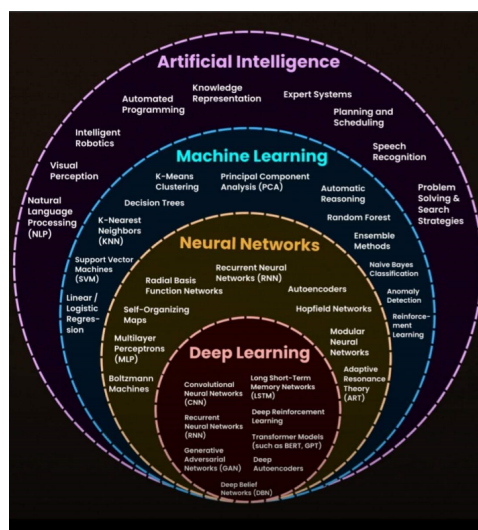


Figure 3.1: The relationship between CNN and deep learning and machine learning

## 3.2 Machine learning

Machine learning is the intersection of computer science and statistical approaches. Machine learning alters or adapts computer operations to improve accuracy. Their correctness is determined by how well the alterations match the intended results. In other words, machine learning employs algorithms to extract features from raw data and convert them into a model, which is then used to make judgments on untested data. Simply said, we want machines to be able to learn from data and eventually become intelligent enough to learn from experiments in the same manner that humans do. Machine learning, as well as the integration of pattern recognition and computer learning theory in artificial intelligence, are constantly evolving. There are two forms of machine learning: supervised and unsupervised. Both types of learning are predicated on the idea that learning is based on both prior experience and testing new material. This new information looks for similarities between data points that can help us learn. ML is also classified based on the sort of product generated..[17]

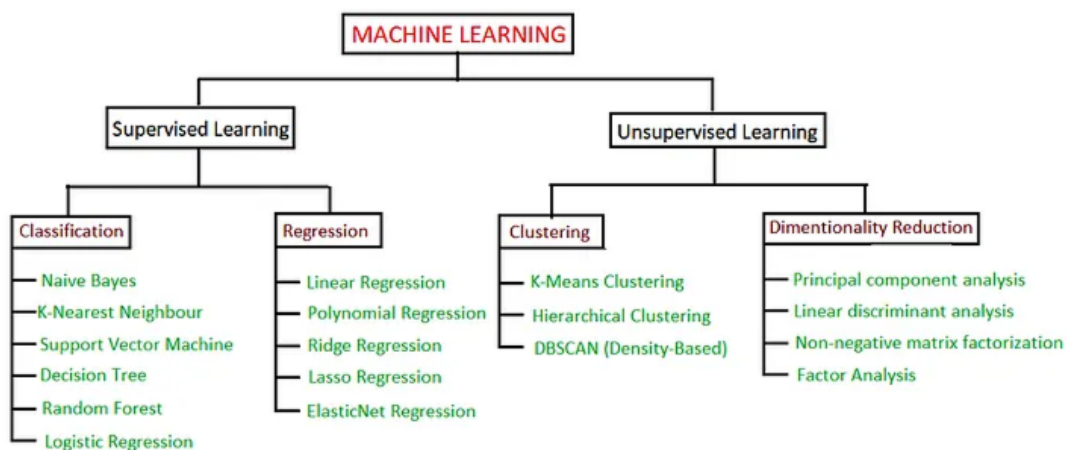


Figure 3.2: Type of Machine learning

### 3.2.1 Supervised learning:

Supervised learning is a machine learning activity that involves learning a function that translates an input to an output using example input-output pairs. It derives a function from labeled training data, which consists of a collection of training samples. Supervised machine learning algorithms are ones that require external assistance. The input dataset is separated into train and test sets. The train dataset contains an output variable that has to be predicted or classified. All algorithms derive patterns from the training dataset and use them to predict or classify the test dataset. The workflow of supervised machine learning algorithms is illustrated in the figure below. The most popular supervised machine learning algorithms have been discussed here.[18]

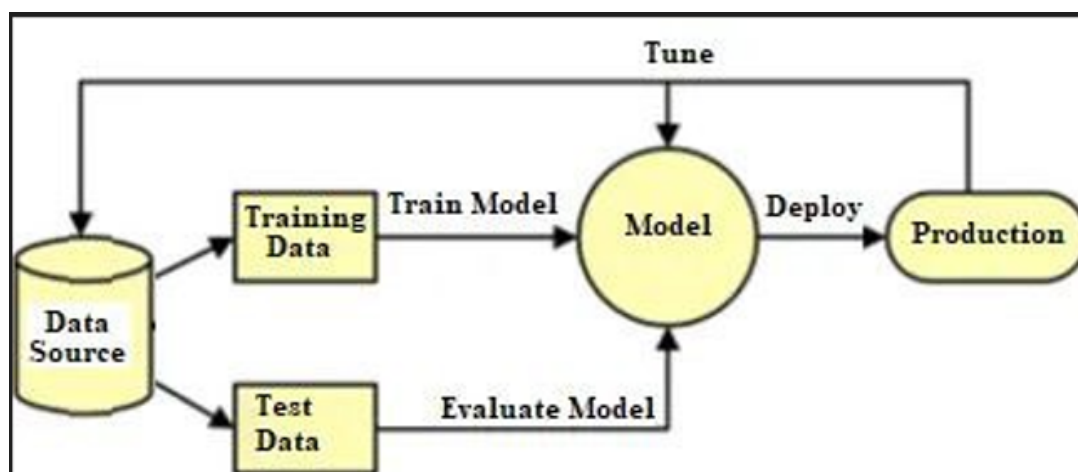


Figure 3.3: Supervised learning Workflow

**Decision Tree:**

A decision tree is a graph that represents options and results in the form of a tree. The graph's nodes represent events or choices, while its edges reflect decision rules or conditions. Each tree has nodes and branches. Each node represents attributes in a group that will be classed, and each branch represents a value that the node can take.[18]

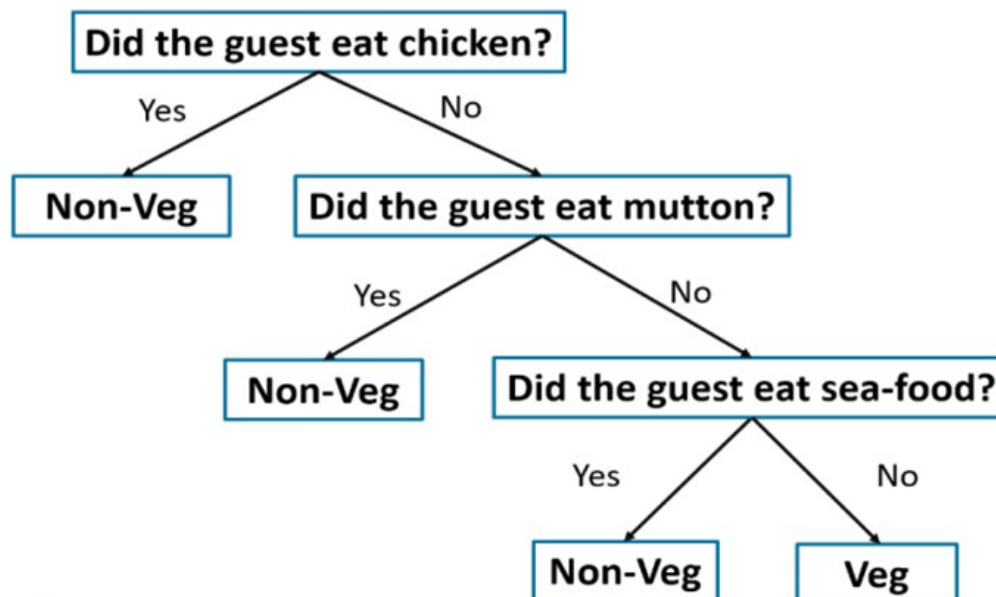


Figure 3.4: Decision Tree

**Support Vector Machine:**

Support Vector Machine (SVM) is another popular modern machine learning approach. Support-vector machines are supervised learning models that use learning techniques to examine data for classification and regression analysis. In addition to linear classification, SVMs may easily

do non-linear classification utilizing the kernel approach, which involves implicitly translating their inputs into high-dimensional feature spaces. It basically creates margins between classes. The margins are drawn in such a way that the distance between the margin and the classes is maximal, hence minimizing the classification error.[18]

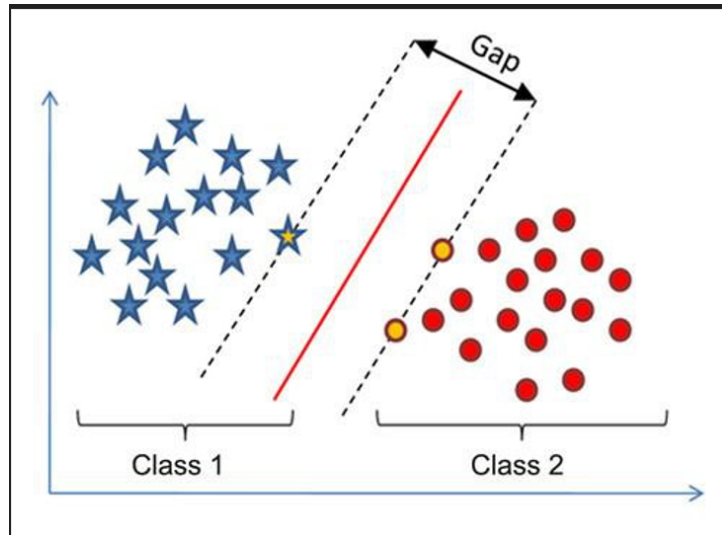


Figure 3.5: Support Vector Machine

### 3.2.2 Unsupervised Learning:

Unsupervised learning is distinguished from supervised learning by the absence of correct answers and a tutor. Algorithms are left to their own devices to uncover and display the fascinating structure in the data. Unsupervised learning algorithms only learn a few features from the data. When new data is introduced, it employs previously learnt features to determine the data's classification. It's primarily utilized for clustering and feature reduction.[18]

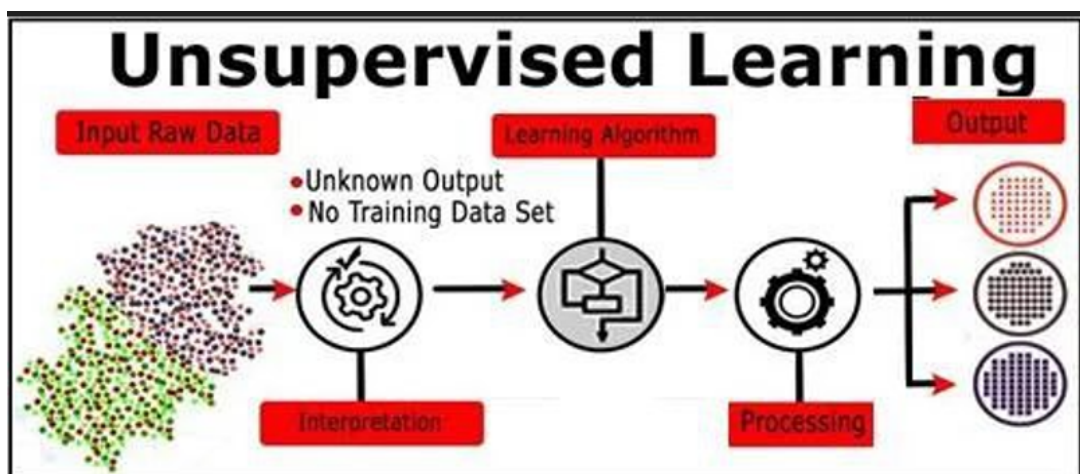


Figure 3.6: Unsupervised Learning



### Principal Component Analysis:

principle component analysis is a statistical process that applies an orthogonal transformation to convert a set of observations of potentially correlated variables into a set of linearly uncorrelated variables known as principle components. This reduces the dimension of the data, making computations faster and easier. It is used to describe the variance-covariance structure of a set of variables using linear combinations. It is commonly used as a dimensionality reduction approach.[18]

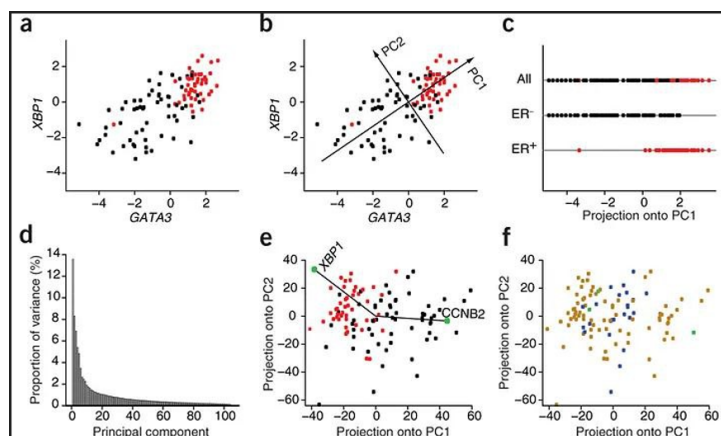


Figure 3.7: Principal Component Analysis

### K-Means Clustering:

K-means is a basic unsupervised learning technique that addresses the well-known clustering problem. The process employs a straightforward approach to classifying a given data set using a predetermined number of clusters. The fundamental idea is to create k centers, one for each cluster. These centers should be located strategically because different locations produce varied results. So the best option is to arrange them as far apart as possible.[18]

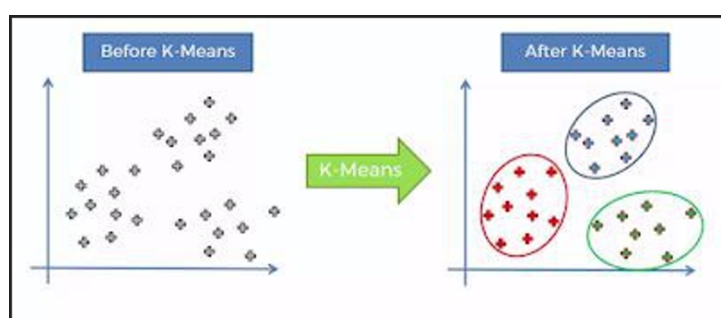


Figure 3.8: K-Means Clustering

The next step is to associate each point in a given data set with the nearest center. When no points remain, the first stage is finished, and an early group age is achieved. At this point, we must recalculate k new centroids to serve as the bary centers of the clusters created in the preceding step.

### 3.3 Neural networks

Neural networks are sophisticated computational models inspired by the human brain's neural structure that are used to handle complex data and perform tasks such as pattern recognition, classification, and regression. These networks are made up of interconnected layers of artificial neurons that work together to convert input data into meaningful results. Learning from labeled examples allows neural networks to generalize patterns and make predictions on previously unseen data, a process known as supervised learning. Neural networks are often used for tasks such as image and speech recognition, natural language processing, and anomaly detection. Neural networks adjust their internal parameters through iterative training with algorithms such as backpropagation to reduce errors and enhance performance on certain tasks. Neural networks' versatility and adaptability make them essential tools in modern artificial intelligence applications, propelling progress in domains as diverse as healthcare, finance, and autonomous systems.

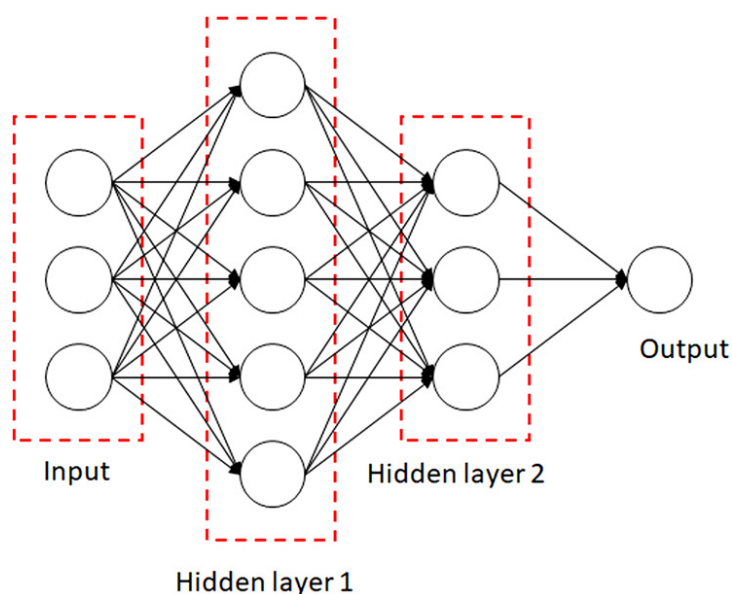


Figure 3.9: A 3-layer neural network with three inputs

#### 3.3.1 Loss Function:

The loss function is an important part of neural network training because it measures model performance by calculating the difference between expected output and actual ground truth labels. In supervised learning, when neural networks are trained on labeled data, the loss function assesses the model's ability to accurately predict outcomes. By assessing the loss, neural networks may assess the quality of their predictions and alter their internal parameters using optimization procedures like backpropagation to eliminate inconsistency. Choosing the right loss function is critical because it directly influences the network's capacity to learn and generalize efficiently. Mean square error is a common form of loss function for regression tasks, while cross entropy loss is used for classification challenges. Ultimately, the optimum goal of

neural networks is to minimize the loss function, which improves the model's prediction skills and overall performance on a variety of tasks.

### 3.3.2 Backpropagation

Backpropagation is an important algorithm in the field of neural networks, as it helps to properly train these complicated models. Backpropagation calculates gradients with respect to network parameters, allowing for iterative modification of weights and biases to reduce prediction errors. Backpropagating error signals from the output layer to the input layer allows neural networks to alter their internal parameters in response to the gradient of the loss function. This process directs network improvement, allowing for improved predictive accuracy and convergence towards optimal solutions during training. Back propagation is crucial for updating model parameters efficiently. It uses the chain rule to calculate gradients layer by layer. This methodological approach assures that neural networks learn from data, adapt to complex patterns, and increase performance on a variety of tasks, including image classification, audio recognition, and natural language processing. Back propagation is fundamental to the success of neural networks because it allows them to learn from data, generate correct predictions, and generalize efficiently across domains.

## 3.4 Deep learning

Before discussing deep learning, it's important to understand the challenges faced by researchers with artificial neural networks. Traditional neural networks often reach a performance plateau, unable to improve despite increasing hidden layers or training data size. Researchers identified and addressed these limitations, leading to the development of deep neural networks, which improve accuracy by increasing layers and augmenting training data. Deep neural networks are categorized into two types: Convolutional Neural Networks (CNNs) and Recursive Neural Networks (RNNs).

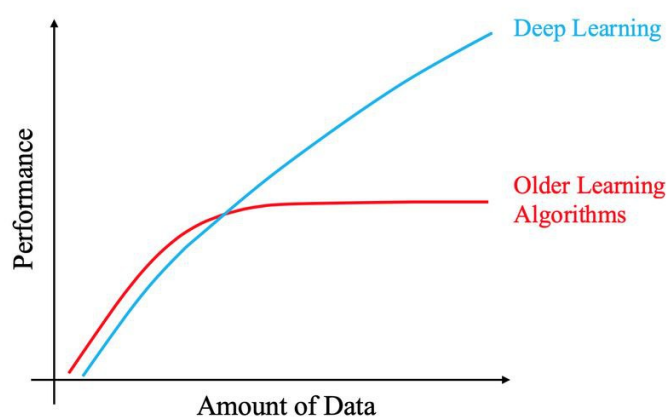


Figure 3.10: Machine learning VS deep learning

### 3.5 convolution neural networks (CNNs)

Convolutional neural networks are one of the types of deep neural networks and are given this name because they contain convolutional layers. It is often used in the fields of computer vision and visual image analysis. They are given this name because they use convolutional layers. It uses the mathematical principle of convolution (convolution). Which refers to the process of combining two jobs and producing a new job from them. Each convolutional layer is made up of arrays of neurons known as activation maps or feature maps. These matrices have both width and height. The depth of this layer is formed by stacking these arrays of neurons. Each of these matrices is associated with a (filter), also known as a (kernel) or a (mask), which is an array of (weights) whose height equals its width and whose size must be less than the size of the image matrix entering the network. This filter is applied horizontally and vertically to the picture matrix's input, requiring one neuron at each location where the filter stops. The final number is derived by multiplying the two matrices. The filter matrix and the matrix above it are part of the image matrix, and the result represents one of the pixels in the feature map matrix. This filter continues to advance down the picture matrix, eventually reaching the grid, where all of the matrix's pixels are tallied. Then comes the candidate's turn. The second is the associated matrix. Until the feature map values are formed, the depth of the convolutional layer is calculated. The training process includes the weights represented by each filter matrix as well as the bias values. A convolutional neural network is made up of layers, each of which alters the previous layer's activations or outputs using a differentiable function. There are other such layers utilized in CNNs, which will be discussed in subsequent sections; however, the most frequent building blocks used in most CNN architectures are the convolution layer, pooling layer, and fully connected layers. These layers are conceptually comparable to feature extractors, dimensional quality reduction layers, and classification layers. The CNN layers are stacked to create a fully convolutional layer. The following figure shows what we said:

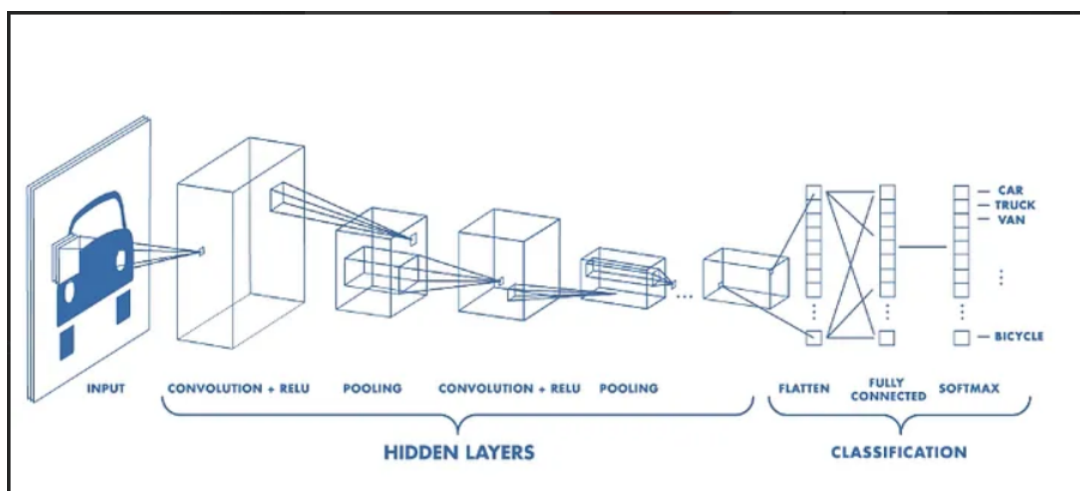


Figure 3.11: CNN-architecture

### 3.5.1 Convolutional Layer

When images are entered into the computer, they are recognized by converting them into a three-dimensional matrix (height x width x depth), and the depth here is equal to (3), which is assigned to each of the main colors. The three red (R), green (G) and blue (B) matrix. The following figure shows these matrices

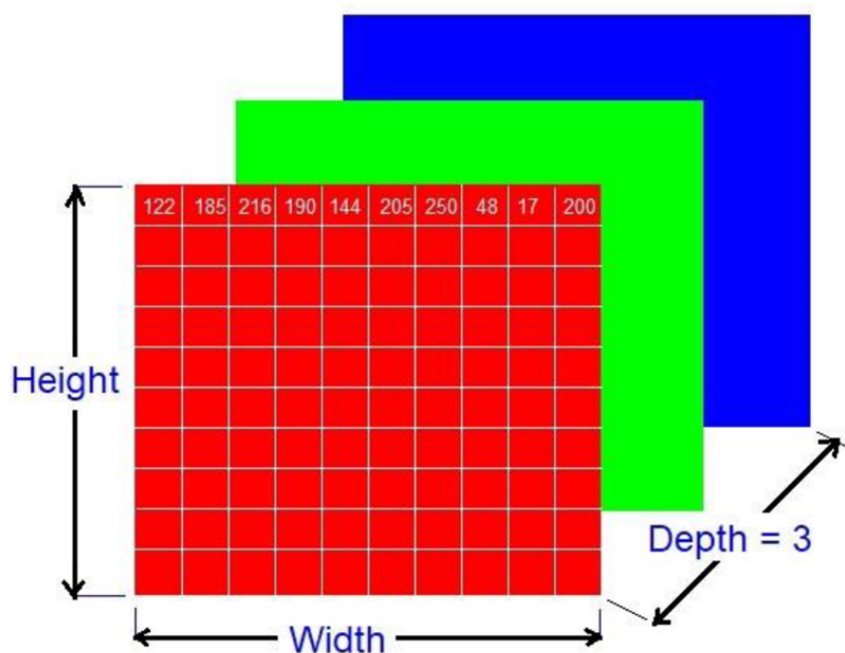


Figure 3.12: 3D data input

In the figure above, we see three matrices. Each matrix represents one of the three main colors. Each matrix contains a group of elements with specific values. Each element represents a point (pixel), as its value represents the amount of illumination for this color that appears when the image is displayed on the screen. The basis of the work of these networks can be explained in the following steps:

1 - The image data is entered as it is, i.e. a three-dimensional matrix, and sometimes the image is in the form of a gray scale with only two dimensions, that is, without converting it to a vector.

2- Small matrices called filters are constructed with the same dimensions as the input image. Each matrix has weights that must be trained. These filters are designed to detect the existence of specific features or characteristics in the original image input data.

3 - The first filter is applied to the image in horizontal and vertical steps until all points are eliminated. Following each movement, the filter matrix's dot product matrix is multiplied by the corresponding part of the image matrix to calculate the value of one point from the first feature matrix associated with the first filter. The computation process is repeated after each movement of the filter matrix until the entire image is scanned and therefore values are computed. The first

characteristic matrix. The second candidate and the second characteristics matrix follow, and so on for the remaining filters and characteristics matrices for each candidate.

4 - Default values are assigned to the filter elements, which, as previously stated, represent weights; these values must differ from one filter to the next in order to acquire various features and characteristics in each matrix of characteristics.

5 - The property matrices as a whole represent input data to the subsequent layer. It then represents the depth of this data.

The following figure shows what we have shown about the convolutional layer:

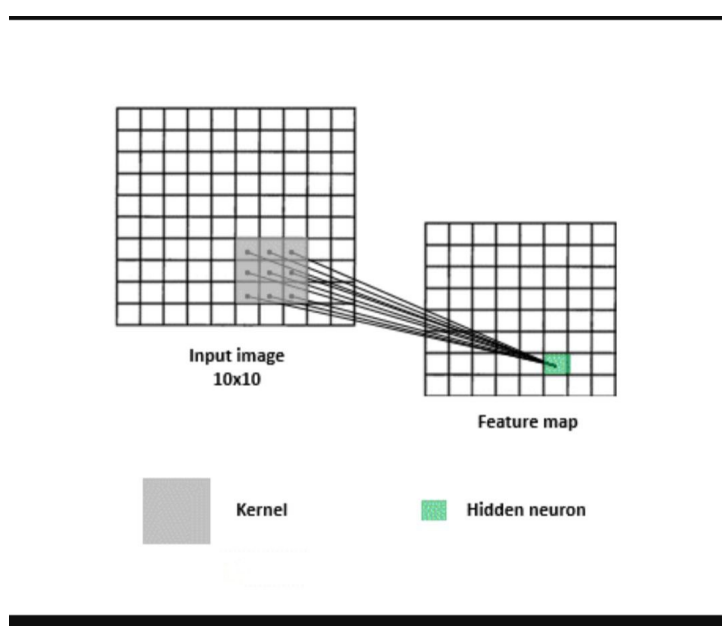


Figure 3.13: feature map

### Stride:

Stride refers to the amount of displacement the kernel experiences as it moves across the input during convolution. A stride of one signifies that the kernel moves one position at a time, whereas a stride of two means that the kernel moves two locations with each movement. Stride has a direct influence on the spatial dimensions of the convolution result. Larger strides can reduce the dimensionality of the output, whereas smaller strides preserve more spatial information. Larger strides minimize computing effort, boosting operational speed, which can have a direct impact on quality. Figure 3.15 shows a kernel in red traversing the image's pixel map with a stride of one.

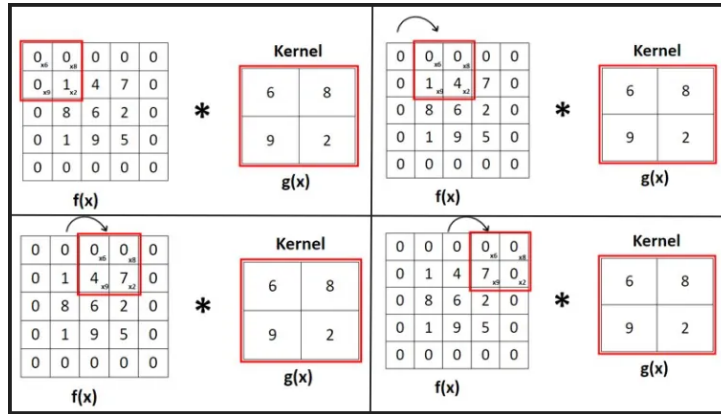


Figure 3.14: Example of Stride

**Padding**

This approach is used to tackle two instances that occasionally exist in CNNs, which are:

- 1 - As the network’s layers progress, the image size gradually lowers.
- 2 - The pixels near the edges are only used in the calculations a few times compared to their counterparts in other regions, reducing the model’s efficiency.

As a result, as a treatment for both scenarios, a frame of pixels with the value (0) is added, and the width of this frame is determined by the filter size employed, if we want the width and height of the feature matrices to be identical to the width and height of the input image. As illustrated in the following figure:

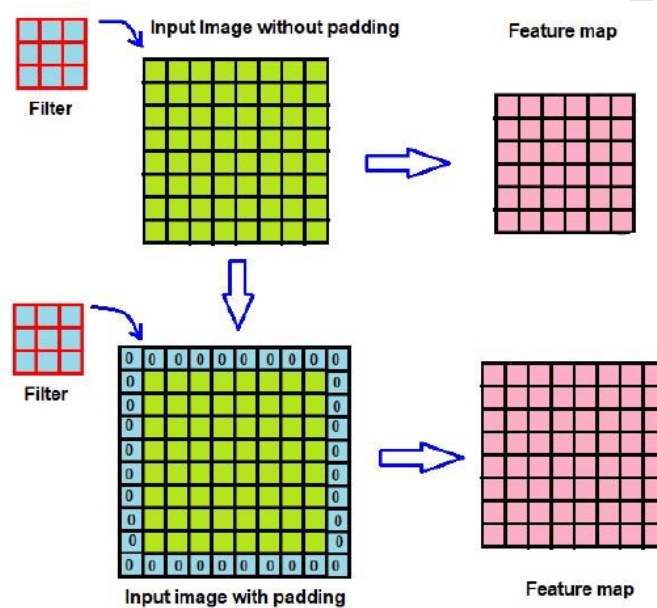


Figure 3.15: padding

## ReLU

The Rectified Linear Unit (ReLU) is a popular nonlinearity used in neural networks, defined as:

$$\text{ReLU}(x) = \max(0, x)$$

The ReLU function returns the input value if it is positive and 0 otherwise. This simple activation function has grown in prominence because to its processing economy and ability to solve the vanishing gradient problem associated with functions such as sigmoid and tanh. In ReLU, the derivative is one for positive inputs and zero for negative inputs. ReLU has been shown to increase convergence during training when compared to classic activation functions such as sigmoid and tanh, particularly in deep neural networks like convolutional neural networks (CNNs). Proper weight initialization and learning rate adjustment are critical when employing ReLU for optimal training and model performance. Variations of ReLU, such as Exponential Linear Units (ELUs), have also been developed to improve the capabilities of this activation function in neural network architectures. The following explains this function:

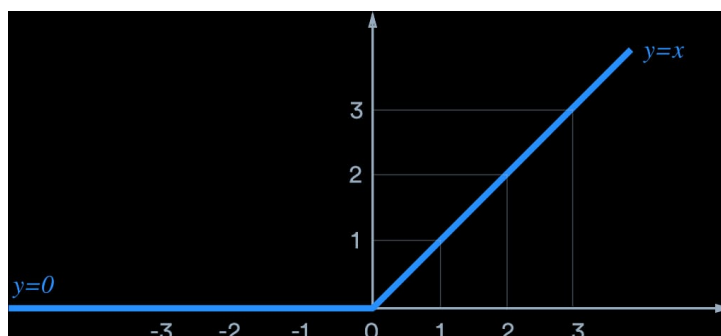


Figure 3.16: RELU

### 3.5.2 Pooling layer

As previously stated, at the convolutional layer, features from the image input to the model are retrieved and stored in feature matrices. One issue here is that these feature matrices are sensitive to the positions of the features in the input image. One way for reducing sensitivity is to minimize the size of the feature matrix, which makes... Feature matrices are stable and resistant to changes in feature positions in the input image. Pooling is used to reduce groups of pixels close to one. There are two approaches of implementing pooling:

- 1- Maximum pooling.
- 2- Average pooling.

A mask of size (2×2) is applied to the feature map matrix with a step of 2. For the first technique, pooling maximum is Select the highest value from the four values shown below for each position of the mask. In the second scenario (average pooling), the average is taken for those values, as explained in the image below.



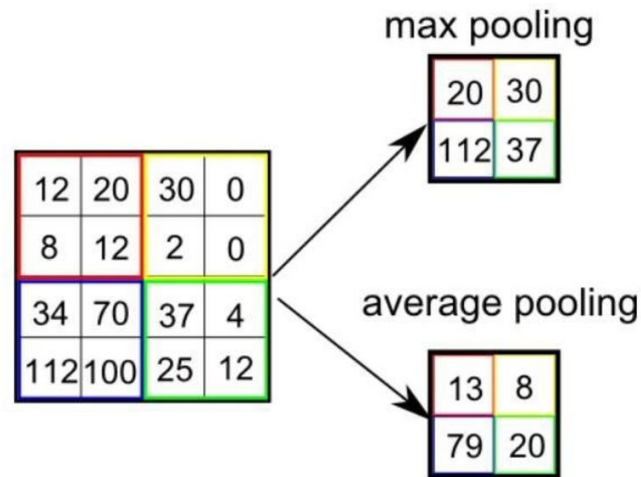


Figure 3.17: polling

This technique is applied to all feature matrices and the results that emerge from them form a new layer called (pooling layer).

### 3.5.3 Fully Connected Layers

It is the layer in which there are no feature matrices, as it consists only of neurons and weights associated with them, as is the case in neural networks of the (ANN) type, and is often used in the last layers in (CNN) networks.

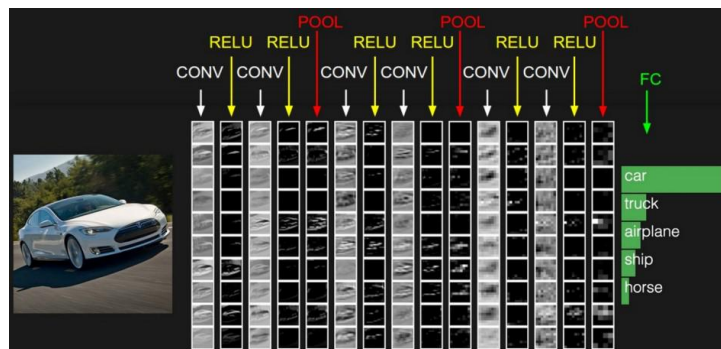


Figure 3.18: Fully Connected Layers

## 3.6 Transfer Learning

Transfer learning is a deep learning technique that trains a CNN rapidly and accurately by importing weights from another CNN rather than starting from scratch. Instead, the weights are imported from another CNN that was trained on a larger dataset. The weights are imported from the ImageNet dataset, which is the most frequent source of weights for transfer learning. Several CNN architectures were trained on the ImageNet dataset and demonstrated remarkable accuracy. Rather than starting from zero, these weights can be used to categorize an entirely

different dataset. There are four ways for transfer learning. The first technique is to remove the original fully connected layer that serves as a classifier, freeze the weights across the whole network, and use the CNN's pre-trained layers for feature extraction. Then, add a classifier layer, such as a fully connected layer, or another machine learning classifier, like a support vector machine. The second technique is to delete the initial fully connected layer, fine-tune the weights of the entire network with a low learning rate (LR), and then add a new classifier layer appropriate for the new task. The third technique is to remove the completely linked layer, fine-tune only the upper layer while freezing the lower layer, and then install a new classifier layer designed for the new task. Many researchers believe that the bottom layer simply recognizes generic features like edges and circles, whereas the upper layer detects more dataset-specific properties. For this reason, numerous authors suggest fine-tuning the top layer. The fourth option is to use cutting-edge architectures and start training from scratch. This entails only using architectures that have been demonstrated to function on a wide range of tough datasets.

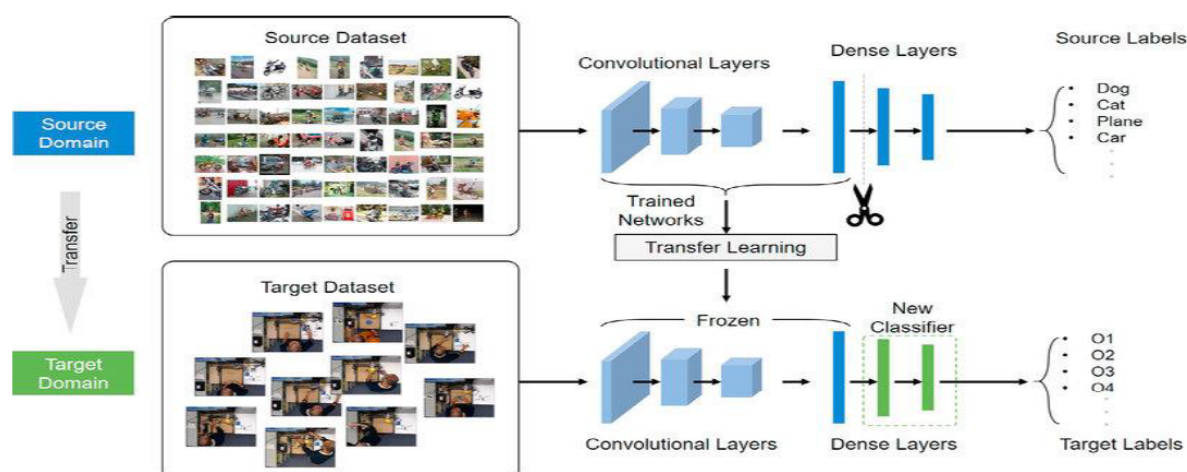


Figure 3.19: The architecture of transfer learning model.

## 3.7 Some of the concepts and techniques used:

### 3.7.1 One-hot encoding

It is a data arrangement strategy used to train networks in categorization and make the process clearer and easier to grasp. This technique converts a portion of the standard output values (Labels) from the expressions that indicate the class into a vector (Vector) with elements equal to the number of classes in the training dataset. For example, suppose we had a group of animal sorts, as shown in the figure below:



Figure 3.20: data set

If we want to use (One-hot encoding) technology, it can be as follows:

Animal	Label
Dog	[1 0 0 0 0]
Fox	[0 1 0 0 0]
Horse	[0 0 1 0 0]
Eagle	[0 0 0 1 0]
Squirrel	[0 0 0 0 1]

Figure 3.21: One-hot encoding

For example, if the image entering the network is (dog), then the (Label) must be the vector (10000). This technique is used with the named activation function(Softmax activation function).

### 3.7.2 Softmax activation function

It is an activation function that is frequently applied after the output layer in deep neural networks used in classification. The values generated by the output layer’s neurons, which are in the form of numbers, are passed into this function, which turns these numbers into a distribution of probabilities such that the total of these probabilities equals (1), and these probabilities are stored in a vector. The following diagram explains the requirement.

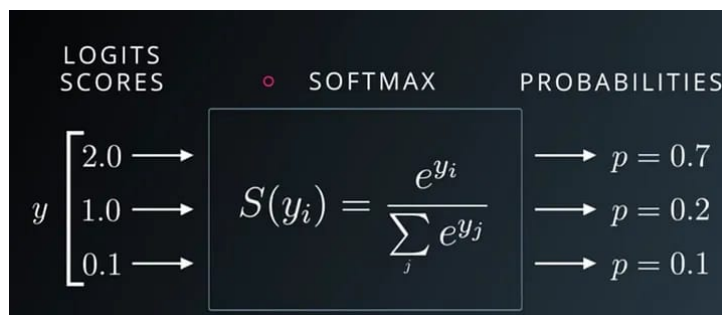


Figure 3.22: softmax

### 3.7.3 Dropout

It is a technique used in deep neural networks to improve training performance by eliminating a negative phenomenon known as overfitting, which can occur as a result of incorrect values in the training data or other factors that will be discussed later. God willing, this procedure will only work during the training phase, when some cells quit operating. These cells are chosen at random in the hidden layers, however this strategy is not employed in the prediction and inference stages because it is unnecessary. The image below illustrates what we stated:

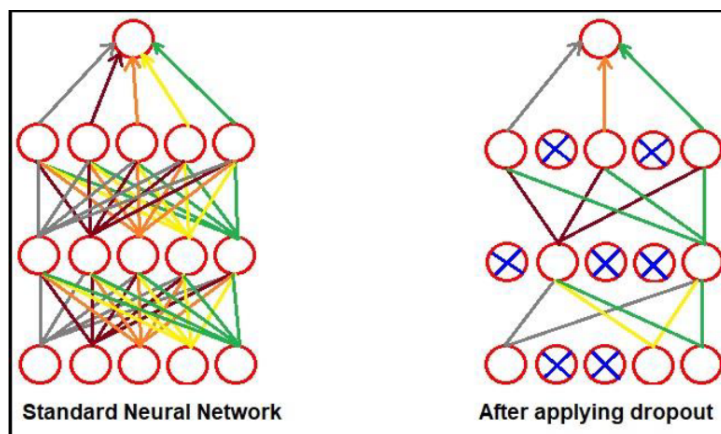


Figure 3.23: dropout

### 3.7.4 Batch normalization

It is a strategy used in deep neural networks with numerous and complex layers to reduce the effect of a phenomena known as (Internal Covariate Shift) that occurs during the training process, resulting in training delay. The output values of the first layer are considered input values for the second layer, and the output values of the second layer are considered input values for the third layer, resulting in a significant increase in the standard deviation. This leads to a substantial divergence in the standard deviation of these values as we advance in the subsequent layers, as the variation takes an increasing trend as we progress in layers, as the next layer is impacted by all the layers that preceding it, and this It has a significant impact on the network's stability and performance throughout training. This technology processes values by calculating the average (0) and standard deviation (1). Using this strategy accelerates the training process, resulting in shorter training times and improved model accuracy. It also reduces the phenomenon of overfitting.

### 3.7.5 Gradient descent

It is a multi-dimensional optimization technique that finds the global, not local, minimum value of the loss function or (Cost function) based on the slope of the function and whether it is in a declining state, as its value in this case is negative. If it is, we are heading in the correct way. In the training phase, a positive slope value indicates that we have exceeded the minimal value

and must return to it. The function's slope is tested after each training step and after updating the weights and bias values (Weights & Biases) throughout network training. This algorithm is one of the famous techniques in the field of machine learning in general and in deep neural networks in particular. The following figure shows the work of this algorithm:

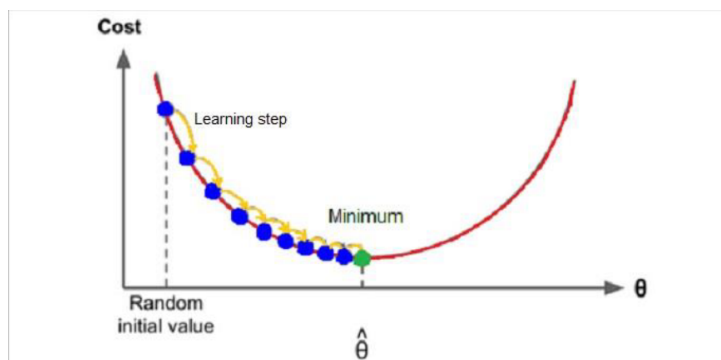


Figure 3.24: Gradient descent

The graph represents the relationship between (Cost) and the value of one of the weights in the network. As the weight value is represented by the blue circles, we started training with a random value for this weight. After that, the weight value is updated at each training step, and the training rate is an important factor in the updating process.

There are three types of this technology: 1-Batch Gradient Descent

2-Stochastic Gradient Descent

3-Mini-batch Gradient Descent

We will explain the second type of these types.

### 3.7.6 Stochastic Gradient Descent(SGD)

The term stochastic refers to a system or process characterized by random probability. As a result, in the (SGD) system, some samples are selected at random rather than (Data set) for each (iteration).

In this technique, the term (Batch) refers to the total number of samples selected (Data set), which is then used to determine the gradient or slope for each iteration. In a typical Gradient Descent algorithm, such as Batch Gradient Descent, the Batch represents the complete dataset. Although using the complete dataset is really valuable for reaching the threshold in a less arbitrary manner, the issue occurs when our datasets become too enormous.

Assume we have a million samples in our dataset. If we use a conventional Gradient Descent optimization strategy, we will need to use all of the million samples to complete one iteration of Gradient Descent, and this must be repeated for each iteration until we reach the minimum. As a result, it becomes quite expensive in terms of performance. This problem is handled using the (SGD) technique, which uses only one sample, i.e. one batch size (batch size=1), to implement each iteration.

### 3.7.7 Flatten layers

After finishing the sequence of convolutional layers and before generating the last layer (full connection layer), all feature matrices are translated into a single vector and linked to the final layer (output layer), which is of the full connection type. That is, this layer acts as a bridge between the two sorts of layers.

## 3.8 Proposed Method:

Figure 3.25 depicts the proposed approach to developing a model for detecting manufacturing defects. The proposed flowchart illustrates a complete approach to creating a manufacturing defect detection model. The process starts with image collecting and progresses to data pre-processing, which includes procedures like data cleaning, cropping, resizing, and annotation. The pre-processed data is then divided into three datasets: training, validation, and testing, to validate the model's generalisability. The training step consists of training and validating the model with training and validation datasets, respectively. The well-tuned manufacturing defect detection model is then run on the test dataset to observe how it performs on an unknown dataset. The recall rate, precision rate, and mAP@0.5 of the model are all assessed during the evaluation phase. This proposed methodology offers a systematic and effective approach to creating a manufacturing defect detection model, which has the potential to significantly improve product quality and efficiency.

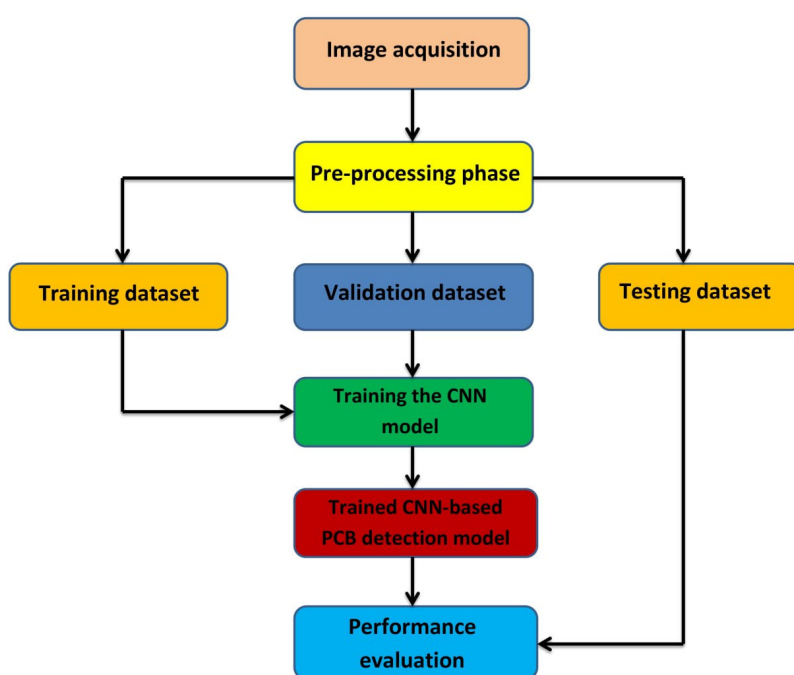


Figure 3.25: Flowchart of the proposed method

### 3.8.1 YOLOv8:

YOLO (You Only Look Once) is one of the most popular modules for real-time object recognition and picture segmentation, and it is now deemed SOTA State-of-the-Art (as of the end of 2023). YOLO is a convolutional neural network that predicts bounding boxes and class probabilities for an image in a single evaluation.

Despite its unquestionable efficiency, it is crucial to remember that this tool was designed for a broad range of uses. However, for more particular applications needing higher quality, speed, and handling of non-standard images, among other scenarios, it is best to understand the architecture and, if necessary, tailor it to meet the task's requirements.

#### **Main Blocks:**

To understand the structure of YOLO, consider that the algorithm is divided into three main blocks, and everything occurs within these blocks: Spine, Neck, and Head. The functions of each block are outlined below.

#### **1-Backbone:**

Function: The backbone, also known as the feature extractor, extracts relevant characteristics from the input. Activities include capturing simple patterns like edges and textures in the initial layers. - Supports many scales of representation, capturing features at various levels of abstraction. - Provides a detailed, hierarchical representation of input.

#### **2-Neck:**

Function: The neck connects the backbone and the head, executing feature fusion processes and incorporating contextual information. Essentially, the Neck assembles feature pyramids by aggregating feature maps obtained by the Backbone; in other words, the Neck gathers feature maps from various phases of the backbone. Activities include concatenating or fusing information of different scales to enable the network to recognize objects of various sizes. - Uses contextual information to improve detection accuracy by taking into account the overall context of the scene. - Reduces spatial resolution and dimensionality of resources to speed up processing, however this can impair model quality.

#### **3-Head:**

Function: The head is the network's final component, responsible for producing outputs such as bounding boxes and confidence scores for object detection. Activities:- Creates bounding boxes for likely items in the image. - Assigns confidence scores to each bounding box, indicating the likelihood of an object's presence. - Sorts the objects in the bounding boxes into their respective categories.

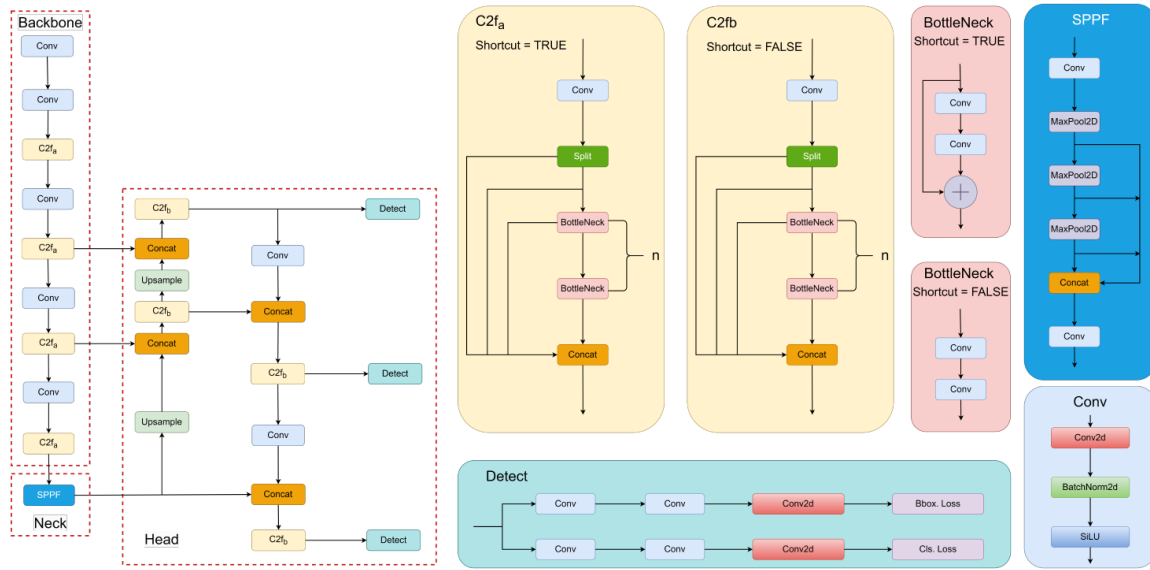


Figure 3.26: YOLOv8 architecture diagram.

### 3.8.2 YOLOv9:

Released in April 2024, YOLOv9 is an open-source model using the YOLOv9 architecture, created by Chien-Yao Wang and his team. YOLOv9 introduces techniques like Programmable Gradient Information (PGI) and Generalized Efficient Layer Aggregation Network (GELAN) to tackle data loss and computational efficiency issues. These innovations ensure YOLOv9 excels in real-time object detection, setting new benchmarks for precision and speed.

#### What is YOLOv9?

YOLOv9 supports object detection and image segmentation, achieving higher mAP than YOLOv8, YOLOv7, and YOLOv5 on the MS COCO dataset. YOLOv9 introduces PGI and GELAN for enhanced efficiency and precision in object detection tasks. PGI addresses data loss in deep networks, retaining crucial features and generating reliable gradients for optimal training. GELAN maximizes parameter utilization and computational efficacy, making YOLOv9 adaptable and high-performing.

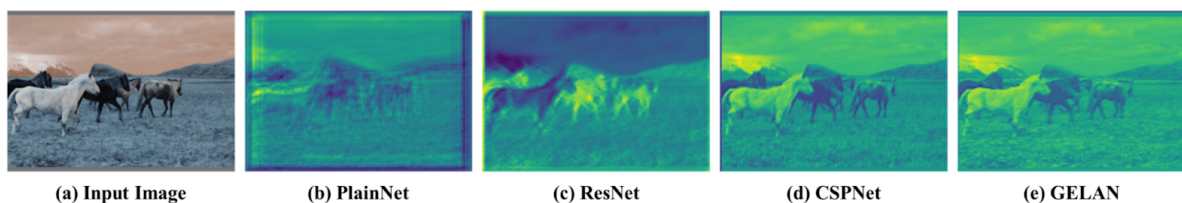


Figure 3.27: Visualization results of random initial weight output feature maps for different network architectures[1]

YOLOv9, which focuses on real-time object identification, uses cutting-edge technologies



like as CSPNet and ELAN, as well as better feature integration techniques, to provide superior performance across a variety of computer vision tasks. YOLOv9 sets a new standard for object detection systems by leveraging PGI's gradient information programming capabilities and GELAN's efficient layer aggregation, exceeding existing real-time detectors in terms of accuracy, speed, and parameter use.

### How YOLOv9 Works

YOLOv9 addresses information loss and network efficiency with four key components: The Information Bottleneck Principle, Reversible Functions, PGI, and GELAN.

#### Background:

The Information Bottleneck Principle. The Information Bottleneck Principle describes the process of information loss as data is transformed within a neural network. This notion, encapsulated in the Information Bottleneck equation, measures the reduction in mutual information between the original and altered data as it moves through the deep network's layers.

$$I(X, X) \geq I(X, f_{\theta}(X)) \geq I(X, g_{\phi}(f_{\theta}(X))), [1] \quad (3.1)$$

This equation expresses mutual information through transformation functions  $f$  and  $g$ , with parameters  $\theta$  and  $\phi$ , respectively. As data  $X$  passes through the layers ( $f_{\theta}$  and  $g_{\phi}$ ) of a deep neural network, it loses crucial information for accurate predictions. This loss may result in unstable gradients and reduce model convergence. Expanding the model can improve data transformation and retain more information. However, this method does not address the problem of unpredictable gradients in very deep networks. The following section explains how reversible functions offer a more practical option. YOLOv9 solves information bottlenecks by utilizing reversible functions, PGI, and GELAN.

#### Reversible Functions

The theoretical remedy to the Information Bottleneck is the Reversible Function. Reversible functions, when embedded in neural networks, ensure that no information is lost during the data transformation process. These functions enable the reversal of data transformations, ensuring that the original input data can be correctly recreated from the network's outputs.

$$X = v_{\zeta}(r_{\psi}(X) \cdot M), [1] \quad (3.2)$$

In the equation above,  $r$  and  $v$  represent the forward and reverse transformations, with parameters  $\psi$  and  $\zeta$ . Using reversible functions allows networks to retain all input information across all layers, leading in more consistent gradient computations for model improvement.

Reversible functions have numerous advantages and represent a departure from the traditional concept of deep networks, particularly when faced with complicated challenges employing models that are not naturally built to be deep.

### Programmable Gradient Information (PGI)

With the advent of reversible functions, there is a need for a new deep neural network training method that not only yields consistent gradients for model updates, but also supports shallow and lightweight neural networks.

Programmable Gradient Information emerges as a solution that combines a primary branch for inference, an auxiliary reversible branch for precise gradient computation, and multi-level auxiliary information to effectively address deep supervision challenges while minimizing inference overheads.

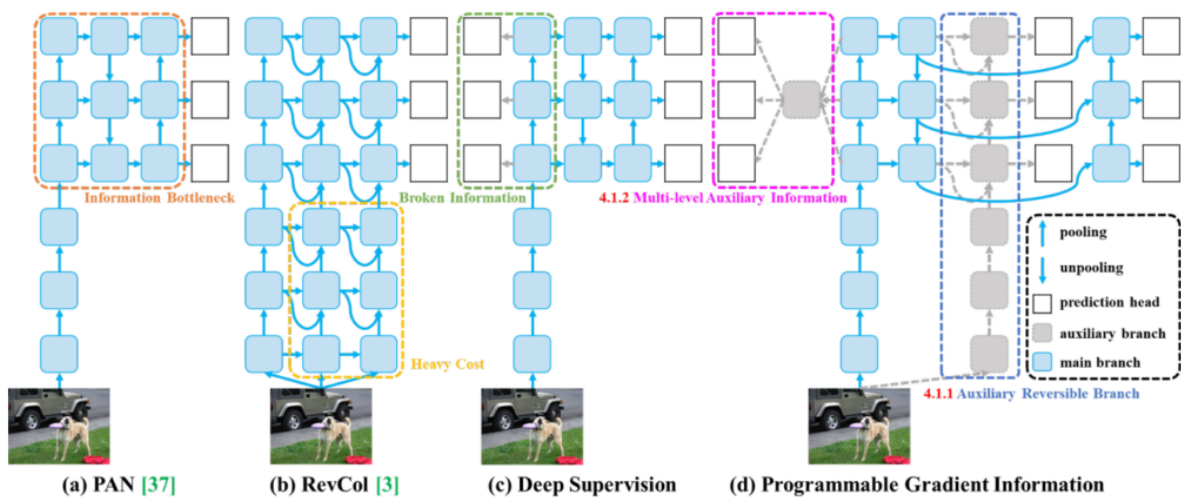


Figure 3.28: Programmable Gradient Information (PGI) Architecture[1]

Examining Programmable Gradient Information within the context of the YOLOv9 architecture reveals how intricately constructed it is to improve model training and performance. PGI promotes precise and effective gradient backpropagation by including an additional supervision node to compensate for the information bottleneck in deep neural networks. PGI is made up of three pieces, each of which has a different but linked purpose in the model's design.

- **Main Branch:** The main branch, which is optimized for inference, makes sure the model runs smoothly and effectively throughout crucial stages. Designed such that no new parts are needed for inference, it maintains great performance without adding to the computational load.

- **Auxiliary Reversible Branch:** The auxiliary branch makes sure that reliable gradients are generated and makes correct parameter changes easier. By utilizing reversible architecture, it reduces the information loss that naturally occurs in deep network layers, maintaining and using all of the data for learning. Because of its flexible design, this branch can be added or removed with ease, guaranteeing that inference speed is not jeopardized by model complexity or depth.

- **Multi-Level Auxiliary Information:** This method uses specialized networks to incorporate gradient information from the model's layers. It solves the difficulty of information loss in deep supervision models by guaranteeing that the model completely comprehends the data. This method improves the forecast accuracy for objects of various sizes.

### Generalized Efficient Layer Aggregation Network (GELAN)

After PGI was incorporated into YOLOv9, there is an obvious need for an even more sophisticated architecture in order to attain the highest level of accuracy. The Generalized Efficient Layer Aggregation Network (GELAN) is a useful tool in this situation.

GELAN presents a unique architecture designed to function in tandem with the PGI framework, improving the model's ability to process and extract insights from data more effectively. While PGI addresses the problem of maintaining critical data across deep neural networks, GELAN expands on this foundation by providing an adaptable and productive framework that can support a variety of computing blocks.

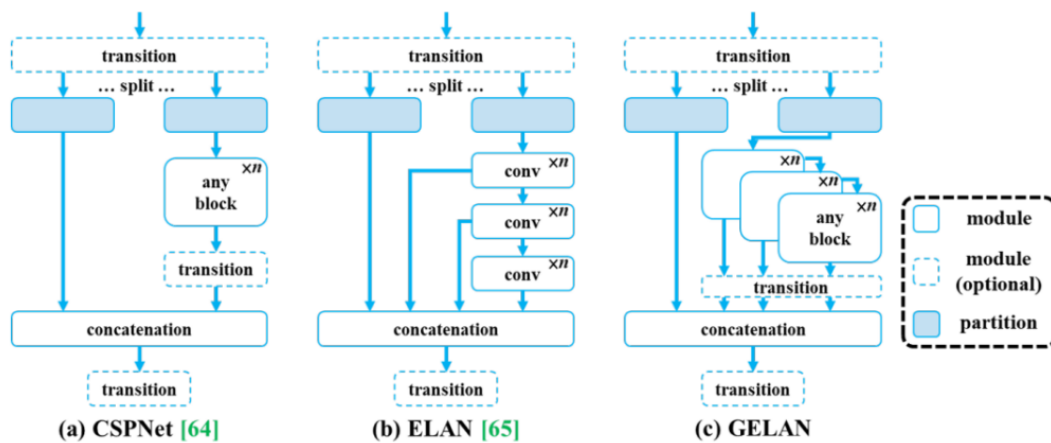


Figure 3.29: Generalized Efficient Layer Aggregation Network (GELAN)[1]

The best features of ELAN's speed optimizations during inference and CSPNet's gradient route planning are combined in YOLOv9 by the GELAN. These features are elegantly integrated by this adaptable architecture, enhancing the YOLO family's renowned real-time inference capabilities. GELAN is a low-weight framework that prioritizes speedy inference while maintaining accuracy, hence increasing the computational blocks' usefulness.

## 3.9 Conclusion:

In this section, we touched on machine learning techniques and its sections and concluded that combining CNN technology, machine learning, and transfer learning provides an advanced and effective solution for detecting defects in manufacturing. By leveraging these advanced technologies, companies can streamline their quality control processes, enhance product quality, and ultimately achieve operational excellence.

## CHAPTER 4

---

# Results and Discussion

---

# Chapter 4

## Results and Descussion

### 4.1 Introduction

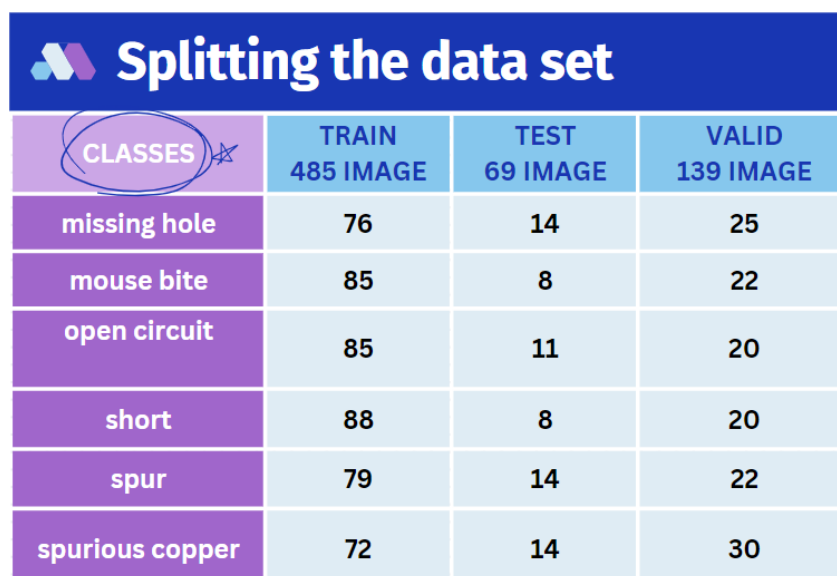
Following feature engineering, dataset selection, and model implementation that resulted in a probability or class output, the metric-based model's performance was evaluated using ensemble test data. The evaluation of deep learning models is an essential component of every project.

### 4.2 Data collection

This work was accomplished in the Google Colab program using T4GPU. We stopped training at 45epoch, upon the expiration of the free period allowed for use in the program. A data set was used for PCB defects obtained from the Robflow [19]. It contains 693 images divided into train test and valid. It contains 6 types of PCB defects are shown in the image below.



Figure 4.1: Input images with dissimilar defects on PCB: missing hole;spurious copper;open circuit;mouse bite;short;spur.



CLASSES	TRAIN 485 IMAGE	TEST 69 IMAGE	VALID 139 IMAGE
missing hole	76	14	25
mouse bite	85	8	22
open circuit	85	11	20
short	88	8	20
spur	79	14	22
spurious copper	72	14	30

Figure 4.2: Splitting the dataset.

### 4.3 Performance Metrics:

Performance metrics are critical tools for determining the accuracy and efficiency of object detection and classification models. They provide insight into how well a model can identify and localize items inside images. They also contribute to a better understanding of how the model handles false positives and negatives. These insights are critical in evaluating and improving the model's performance. We'll look at the various performance measures related with YOLOv8 and YOLOv9, as well as their importance and interpretation.

Let's start by discussing some metrics that are not only important to YOLO but are broadly applicable across different models.

- **Intersection over Union (IoU):** IoU is a metric that quantifies the overlap between a predicted bounding box and the ground truth bounding box. It is critical for determining object localization accuracy.

- **Average Precision (AP):** AP calculates the area under the precision-recall curve, which yields a single value that summarizes the model's precision and recall performance.

- **Mean Average Precision (mAP):** mAP builds on the concept of AP by determining the average AP values over various object classes. This is useful in multi-class object identification scenarios to provide a thorough evaluation of the model's performance.

- **Precision and Recall:** Precision is the fraction of genuine positives out of all positive predictions, evaluating the model's ability to prevent false positives. Recall, on the other hand, estimates the proportion of true positives out of all actual positives, indicating the model's capacity to recognize all instances of a class.

- **F1 Score:** The F1 Score is the harmonic mean of precision and recall, which provides a balanced evaluation of a model's performance while accounting for both false positives and false negatives.

## 4.4 Results :

Below are the details of the visual output obtained by applying yolov8 and yolov9 to the used dataset

- **F1 Score Curve :**

This curve shows the F1 score at various thresholds. Interpreting this curve can provide insight into the model’s balance of false positives and false negatives at various levels.

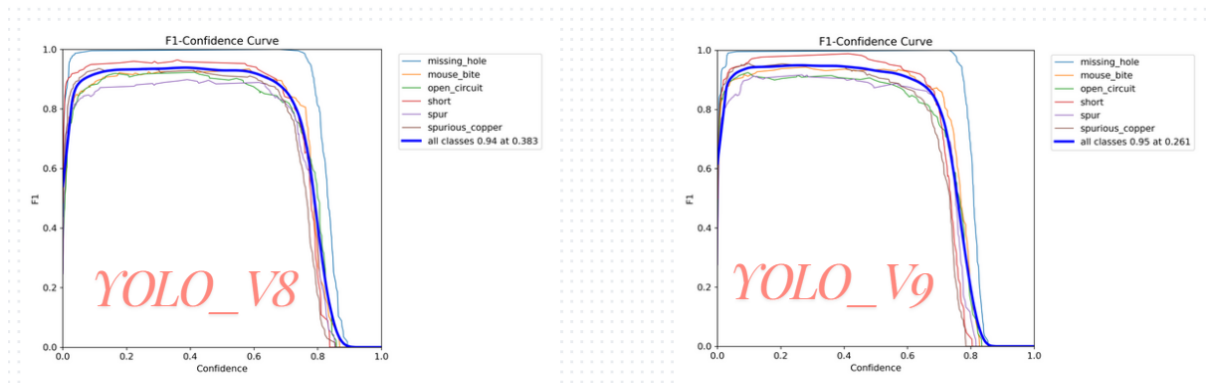


Figure 4.3: F1 Score Curve

- **Precision-Recall Curve :**

This curve, which is an essential picture for any classification task, depicts the trade-offs between precision and recall at various threshold levels. It is especially important when dealing with uneven classes.

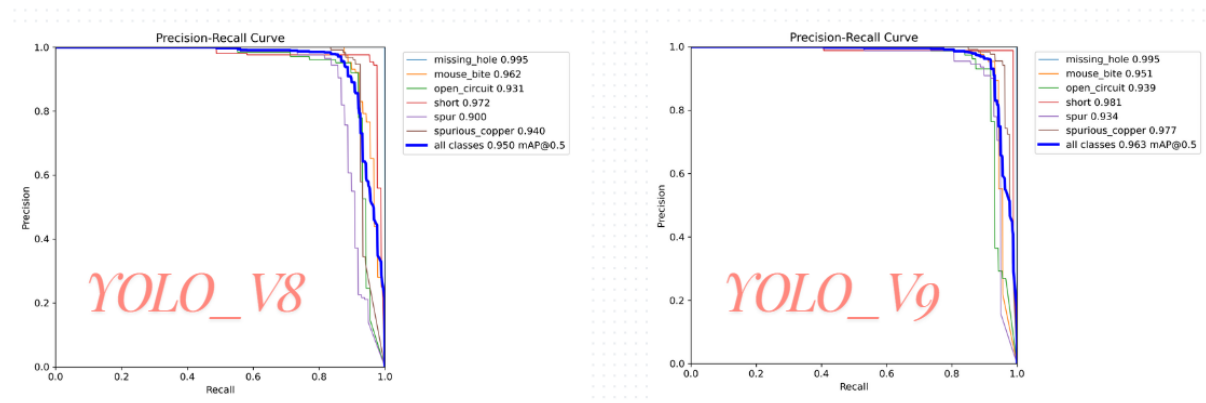


Figure 4.4: Precision-Recall Curve

- **Precision Curve :** A graphical representation of precision values at different thresholds. This curve helps in understanding how precision varies as the threshold changes.

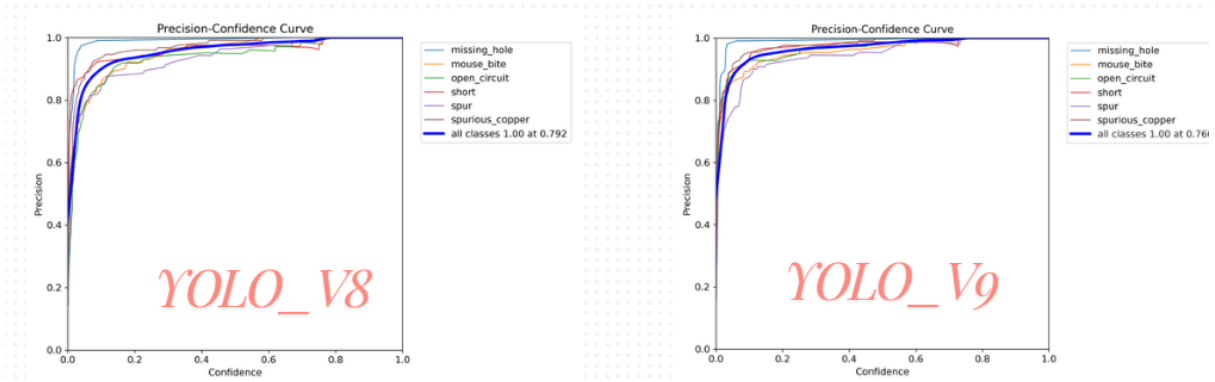


Figure 4.5: Precision Curve

• **Recall Curve** : Similarly, this graph depicts how recall values vary over different thresholds.

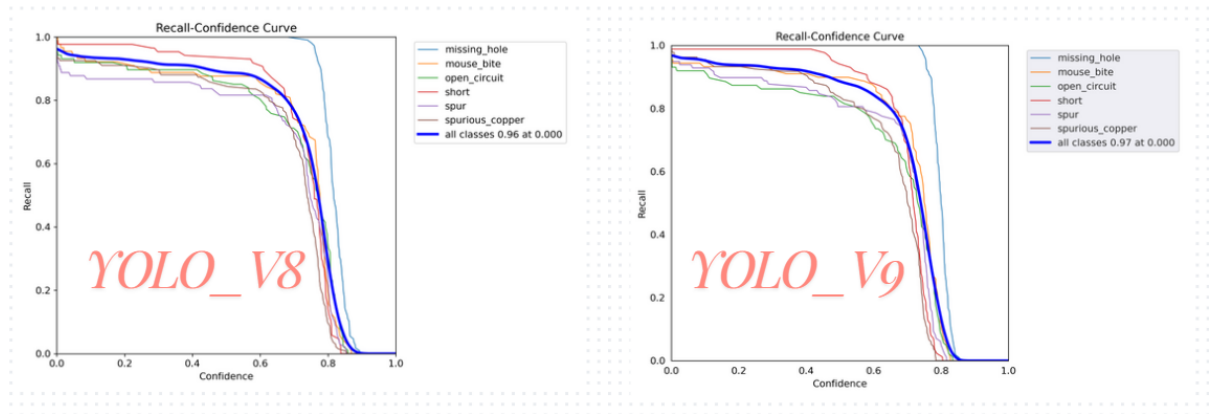


Figure 4.6: Recall Curve

• **Confusion Matrix** : The confusion matrix shows a detailed picture of the results, including the number of true positives, true negatives, false positives, and false negatives for each class.

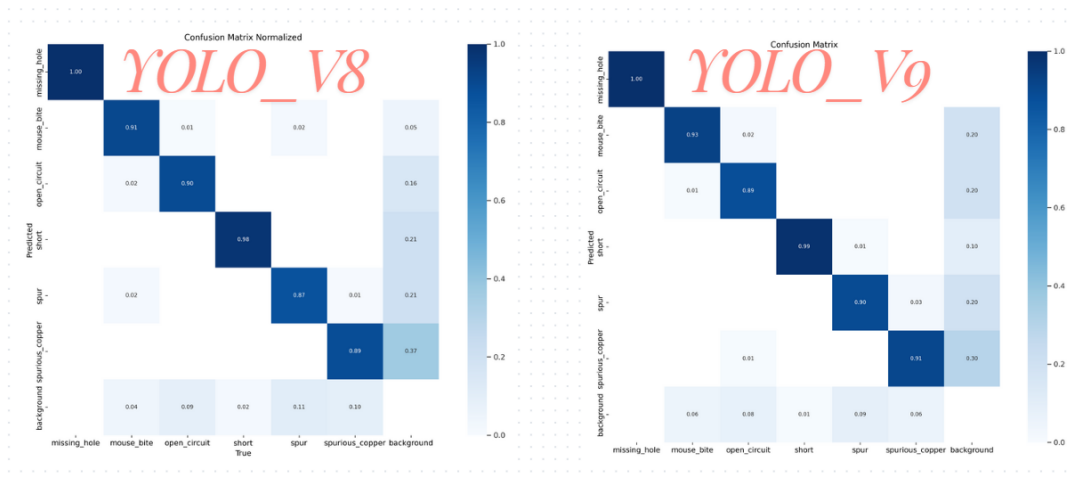


Figure 4.7: Confusion Matrix



• **Validation Batch Labels :** These images depict the ground truth labels for distinct batches from the validation dataset. They provide a clear picture of what the objects are and their respective locations as per the dataset.



Figure 4.8: Validation Batch Labels YOLOv8

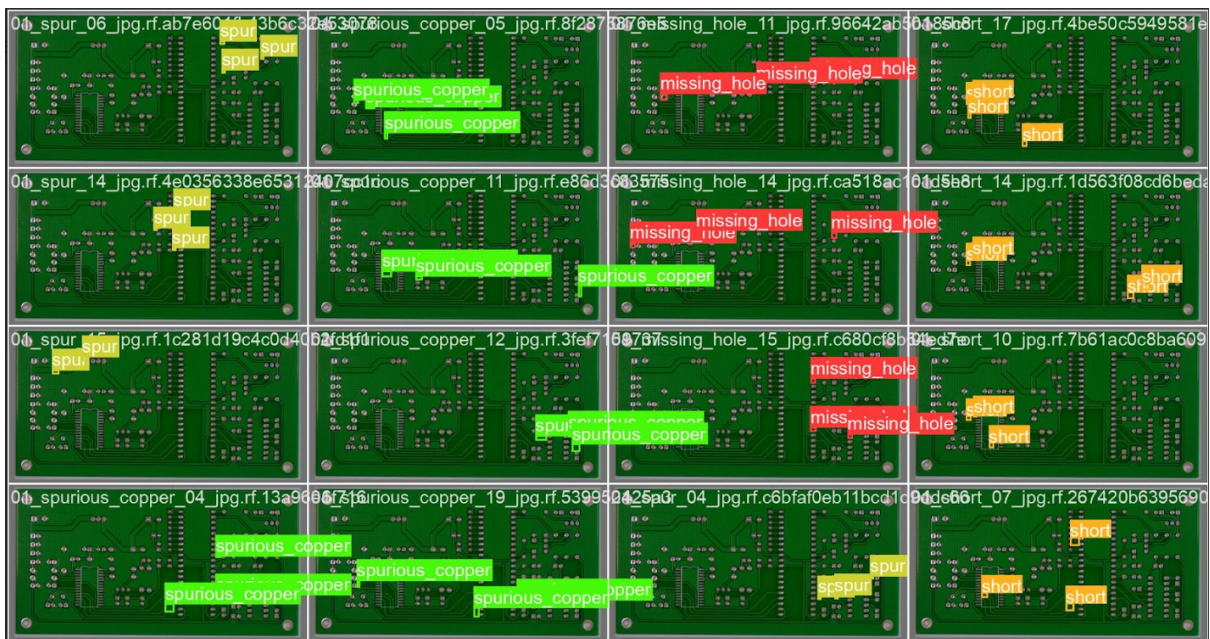


Figure 4.9: Validation Batch Labels YOLOv9

• **Validation Batch Predictions :** Contrasting the label images, these visuals display the predictions made by the YOLO model for the respective batches. By comparing these to the label images, you can easily assess how well the model detects and classifies objects visually.



Figure 4.10: Validation Batch Predictions YOLOv8

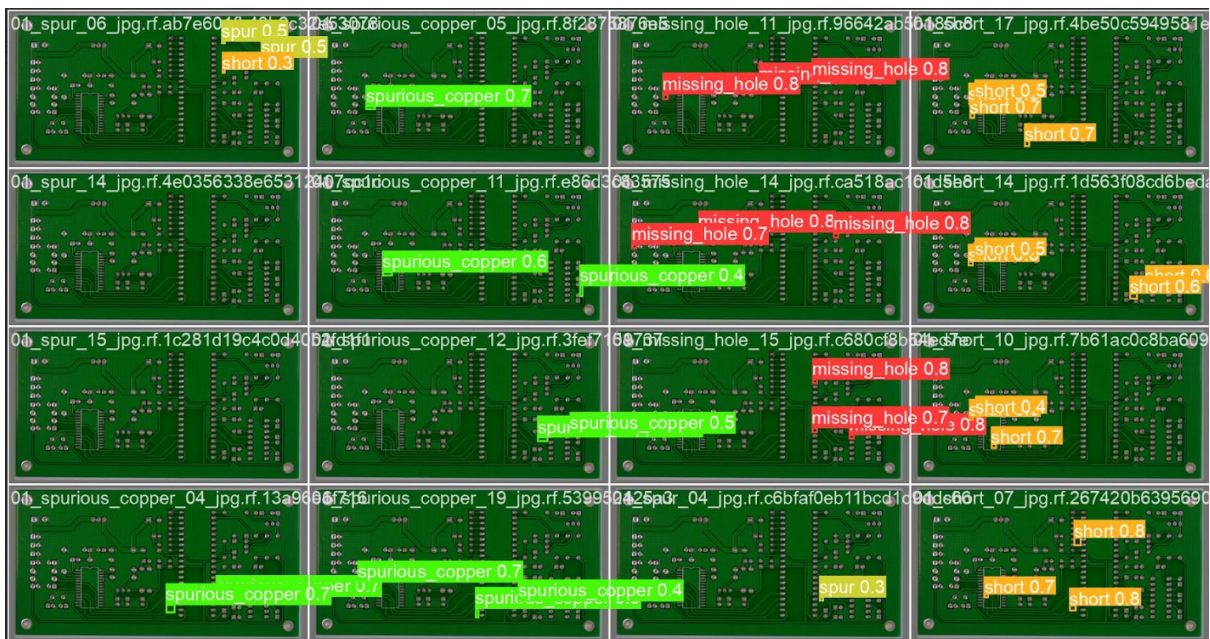
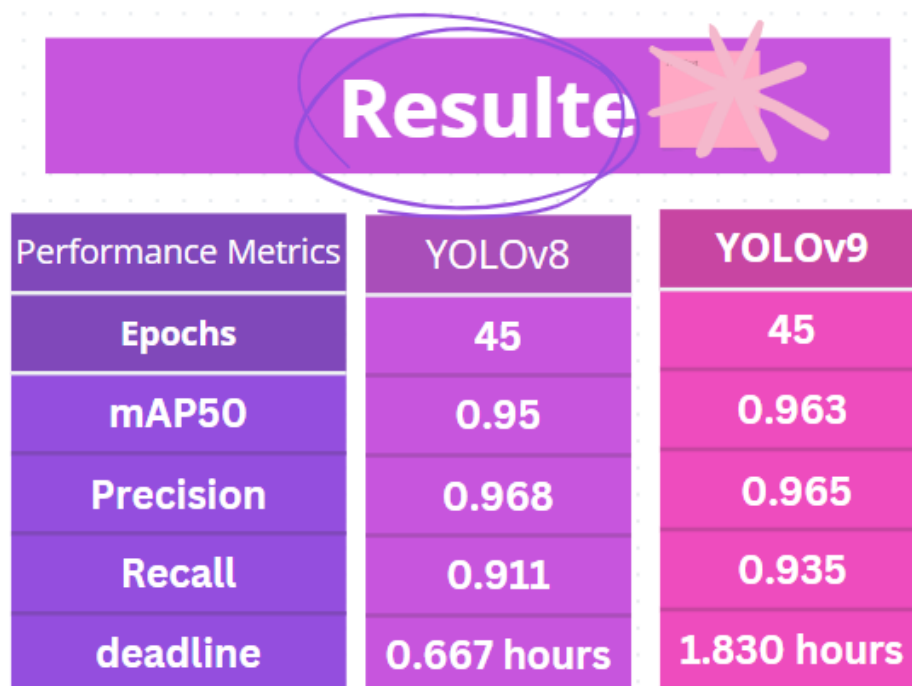


Figure 4.11: Validation Batch Predictions YOLOv9

## 4.5 Descussion:

By training the data set on yolov9 and yolov8, we obtained results that were displayed above, and we were able to summarize them in the following Figure:



Performance Metrics	YOLOv8	YOLOv9
Epochs	45	45
mAP50	0.95	0.963
Precision	0.968	0.965
Recall	0.911	0.935
deadline	0.667 hours	1.830 hours

Figure 4.12: Results

The table provided shows the results of the test we tried. Testing was performed using two different anomaly detection methods: YOLOv8 and YOLOv9.

The data in the table compares the performance of the YOLOv8 and YOLOv9 methods across four benchmarks:

**MAP50:** This metric represents the average square accuracy at 50 threshold levels. It refers to the model's accuracy in accurately locating objects. YOLOv9 performed significantly better than YOLOv8, achieving 0.963 versus 0.95, respectively.

**Accuracy:** Accuracy represents the ratio of correctly detected objects to the total number of objects in the image. YOLOv9 performed slightly better than YOLOv8, achieving 0.965 versus 0.968 respectively.

**Recall:** Recall is the ratio of actually detected objects to the total number of objects in the image. YOLOv8 performed significantly better than YOLOv9, achieving 0.935 versus 0.911 respectively.

**Time taken:** Time taken represents the time required to complete the object detection process, as YOLOv8 took a much shorter time than YOLOv9, taking half an hour compared to YOLOv9, which took approximately two hours.

Overall, YOLOv9 showed better performance than YOLOv8 in terms of accuracy and execution time. However, YOLOv9 showed slightly better performance in terms of recall. The

test parameters of the two models can be modified to improve their performance. If accuracy and performance are the top priority, YOLOv8 is the best choice. If speed is the top priority, YOLOv8 is the best choice.

Model performance also depends on the dataset used for training. Performance metrics test results may differ if a different data set is used. The speed of the form may also vary depending on the computer used.

## **4.6 Conclusion:**

Overall, the effectiveness of the deep learning model in identifying PCB defects is highlighted in this chapter and the importance of careful evaluation in evaluating model performance is emphasized. The visualizations and metrics presented in this chapter provide valuable insights into the model's capabilities and provide a solid foundation for further research and improvements in defect detection processes.

## CHAPTER 5

---

### **General Conclusion:**

---

# Chapter 5

## General Conclusion:

### 5.1 Conclusion:

In this thesis, we identified convolutional neural networks (CNNs) to detect defects in manufacturing. We touched on a range of key areas including the fundamentals of machine learning, neural networks, deep learning, and specific techniques used within CNNs such as convolutional layers, pooling layers, and fully connected layers. We also discussed advanced topics such as transfer learning and presented a proposed method incorporating YOLOV8 and YOLOV9 models for anomaly detection.

We highlight the effectiveness of CNNs in accurately identifying defects in various manufacturing processes. By taking advantage of high-resolution imaging and advanced neural network architecture. CNNs enable us to detect and classify defects with high accuracy, thus enhancing quality control in manufacturing settings.

In this thesis we apply convolutional neural networks (CNNs) to detect defects in manufacturing processes, with a particular focus on identifying PCB defects using deep learning techniques. The dataset was selected, model implementation, and evaluation using ensemble test data to evaluate model performance. Visualizations such as precision curves, recall curves, and confusion matrices were used to analyze the model's accuracy, recall, and predictive capabilities across different defect classes.

#### **Future trends:**

Looking to the future, the field of defect detection in manufacturing using CNNs is poised for several developments: Hybrid models: Combining CNNs with other AI techniques such as reinforcement learning and generative adversarial networks (GANs) can lead to more powerful anomaly detection systems. Edge computing: Deploying CNN models on edge devices can facilitate real-time anomaly detection without relying on central computing resources, thus reducing latency.

Improved Data Augmentation: Innovative data augmentation techniques will continue to improve model training, making defect detection systems more adaptable to different manufac-

turing conditions. Sustainability and Efficiency: AI-based defect detection will play a critical role in creating more sustainable manufacturing processes by reducing waste and improving resource efficiency. In conclusion, although significant progress has been made in the use of CNNs for defect detection in manufacturing, continued research and development are necessary to overcome current limitations and fully realize the potential of these techniques in industrial applications.

## References

- [1] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, “Yolov9: Learning what you want to learn using programmable gradient information,” *arXiv preprint arXiv:2402.13616*, 2024.
- [2] M. P. Samy, S. Foong, G. S. Soh, and K. S. Yeo, “Automatic optical & laser-based defect detection and classification in brick masonry walls,” in *2016 IEEE Region 10 Conference (TENCON)*, pp. 3521–3524, IEEE, 2016.
- [3] F. Alenezi, A. Armghan, S. N. Mohanty, R. H. Jhaveri, and P. Tiwari, “Block-greedy and cnn based underwater image dehazing for novel depth estimation and optimal ambient light,” *Water*, vol. 13, no. 23, p. 3470, 2021.
- [4] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, “Deep learning for smart manufacturing: Methods and applications,” *Journal of manufacturing systems*, vol. 48, pp. 144–156, 2018.
- [5] F. S. Alenezi and S. Ganesan, “Geometric-pixel guided single-pass convolution neural network with graph cut for image dehazing,” *IEEE Access*, vol. 9, pp. 29380–29391, 2021.
- [6] X. Tao, Z. Wang, Z. Zhang, D. Zhang, D. Xu, X. Gong, and L. Zhang, “Wire defect recognition of spring-wire socket using multitask convolutional neural networks,” *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 8, no. 4, pp. 689–698, 2018.
- [7] F. Alenezi, “Image dehazing based on pixel guided cnn with pam via graph cut.,” *Computers, Materials & Continua*, vol. 71, no. 2, 2022.
- [8] V. A. Adibhatla, H.-C. Chih, C.-C. Hsu, J. Cheng, M. F. Abbod, and J.-S. Shieh, “Defect detection in printed circuit boards using you-only-look-once convolutional neural networks,” *Electronics*, vol. 9, no. 9, p. 1547, 2020.
- [9] E. Zhang, B. Li, P. Li, and Y. Chen, “A deep learning based printing defect classification method with imbalanced samples,” *Symmetry*, vol. 11, no. 12, p. 1440, 2019.
- [10] J. P. Yun, W. C. Shin, G. Koo, M. S. Kim, C. Lee, and S. J. Lee, “Automated defect inspection system for metal surfaces based on deep learning and data augmentation,” *Journal of Manufacturing Systems*, vol. 55, pp. 317–324, 2020.



- [11] A. N. Beskopylny, E. M. Shcherban', S. A. Stel'makh, L. R. Mailyan, B. Meskhi, I. Razveeva, A. Kozhakin, D. El'shaeva, N. Beskopylny, and G. Onore, "Discovery and classification of defects on facing brick specimens using a convolutional neural network," *Applied Sciences*, vol. 13, no. 9, p. 5413, 2023.
- [12] Y.-C. Huang, K.-C. Hung, and J.-C. Lin, "Automated machine learning system for defect detection on cylindrical metal surfaces," *Sensors*, vol. 22, no. 24, p. 9783, 2022.
- [13] D. Zhan, R. Huang, K. Yi, X. Yang, Z. Shi, R. Lin, J. Lin, and H. Wang, "Convolutional neural network defect detection algorithm for wire bonding x-ray images," *Micro-machines*, vol. 14, no. 9, p. 1737, 2023.
- [14] J. Yang, S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang, "Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges. materials 13 (24): 5755," 2020.
- [15] H.-Y. Chen, C.-C. Lin, M.-H. Horng, L.-K. Chang, J.-H. Hsu, T.-W. Chang, J.-C. Hung, R.-M. Lee, and M.-C. Tsai, "Deep learning applied to defect detection in powder spreading process of magnetic material additive manufacturing," *Materials*, vol. 15, no. 16, p. 5662, 2022.
- [16] Z. Zhang, G. Wen, and S. Chen, "Weld image deep learning-based on-line defects detection using convolutional neural networks for al alloy in robotic arc welding," *Journal of Manufacturing Processes*, vol. 45, pp. 208–216, 2019.
- [17] I. Aziz, "Deep learning: an overview of convolutional neural network (cnn)," 2020.
- [18] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.
- [19] hamid sakat, "Xor dataset." <https://universe.roboflow.com/hamid-sakat/xor>, apr 2022. visited on 2024-06-01.