

**CLASSIFICATION OF SEMICONDUCTOR WAFER MIXED-
TYPE DEFECTS WITH DEEP LEARNING**

LOW BOON KIAT

**FACULTY OF COMPUTER SCIENCE AND INFORMATION
TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR**

2024

CLASSIFICATION OF SEMICONDUCTOR WAFER MIXED-TYPE
DEFECTS WITH DEEP LEARNING

LOW BOON KIAT

RESEARCH REPORT SUBMITTED TO THE FACULTY OF
COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITI MALAYA, IN PARTIAL FULFILMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF DATA
SCIENCE

2024

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Low Boon Kiat (I.C/Passport No: 970412-10-6387)

Registration/Matric No: 17138399

Name of Degree: Master of Data Science

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Classification of Semiconductor Wafer Mixed-Type Defects with Deep Learning

Field of Study: Image Processing

I do solemnly and sincerely declare that:

- (1) I am the sole author/writer of this Work;
- (2) This Work is original;
- (3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
- (4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
- (5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
- (6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Boon Kiat

18 June 2024

Candidate's Signature

Date

Subscribed and solemnly declared before,

Witness's Signature



Date 19/6/2024

Name: **Profesor Madya Dr. Aznul Qalid Md Sabri**
Pengarah
Designation: **Pusat Inovasi & Perusahaan UM (UMCIE)**
Universiti Malaya

CLASSIFICATION OF SEMICONDUCTOR WAFER MIXED-TYPE DEFECTS WITH DEEP LEARNING

ABSTRACT

Wafer bin map (WBM) defect patterns are crucial for root cause analysis in semiconductor manufacturing, which directly impacts product yield. Traditionally, defect monitoring has relied on manual inspection, which is inefficient and prone to human error. To address these issues, the industry has shifted towards leveraging machine learning for automating wafer defect monitoring. However, conventional machine learning approaches often fall short due to their reliance on manual feature extraction, which is computationally intensive and can lead to information distortion. In contrast, deep learning, particularly Convolutional Neural Networks (CNNs), offers a promising solution for efficient pattern recognition without manual feature extraction. Despite the advancements, most of the published literatures focus on single-type defect classification. In practical industry scenarios, a defective wafer often contains more than one type of defects. Accurately identifying all types of defects on a wafer is necessary for effective root cause analysis. Currently, only a limited number of studies have addressed the classification of mixed-type defects. Additionally, majority of these studies focused on detecting a maximum of two defects per wafer. This approach falls short of meeting the industry's needs for a comprehensive detection system that can identify all defect patterns present on a wafer. Given the limited number of research on mixed-type defect classification and the limitations of existing studies, a significant research gap remains in applying deep learning techniques, specifically CNNs, to classify mixed-type defect patterns on WBMs effectively. This research employed the recent iteration of the YOLO architecture, YOLOv8, to classify both single-type and mixed-type defects across a complex dataset, MixedWM38 dataset, which includes 38 classes of WBM defect patterns. The findings demonstrate that YOLOv8 models, particularly the YOLOv8l

variant, demonstrated improvements in classification accuracy and processing speed, achieving up to 98.4% accuracy and an inference time of 0.7 milliseconds. These results surpass the performance of existing deep learning models. Future research should focus on enhancing model robustness and ensuring equitable performance across all classes by augmenting data for underrepresented classes and employing resampling techniques to balance the class distribution.

Keywords: Semiconductor manufacturing, wafer bin map defect classification, deep learning, YOLOv8

KELASIFIKASI KECACATAN JENIS CAMPURAN WAFER SEMIKONDUKTOR DENGAN PEMBELAJARAN MENDALAM

ABSTRAK

Pola kecacatan peta bin wafer (WBM) adalah penting untuk analisis punca akar dalam pengeluaran semikonduktor, yang secara langsung mempengaruhi hasil produk. Secara tradisional, pemantauan kecacatan bergantung pada pemeriksaan manual, yang tidak efisien dan cenderung kepada kesilapan manusia. Untuk menangani isu-isu ini, industri telah beralih ke penggunaan pembelajaran mesin untuk mengautomatiskan pemantauan kecacatan wafer. Walau bagaimanapun, pendekatan pembelajaran mesin konvensional sering kali tidak memadai kerana bergantung pada pengambilan ciri manual, yang memakan masa komputasi dan boleh menyebabkan distorsi maklumat. Sebaliknya, pembelajaran mendalam, khususnya Rangkaian Neural Konvolusi (CNN), menawarkan penyelesaian yang menjanjikan untuk pengenalan corak yang efisien tanpa pengambilan ciri manual. Walaupun adanya kemajuan, kebanyakan literatur yang diterbitkan memberi tumpuan pada pengelasan kecacatan jenis tunggal. Dalam senario industri yang praktikal, wafer yang cacat sering mengandungi lebih daripada satu jenis kecacatan. Mengenal pasti semua jenis kecacatan pada wafer adalah perlu untuk analisis punca akar yang berkesan. Saat ini, hanya sebilangan kecil kajian yang telah menangani pengelasan kecacatan jenis campuran. Tambahan pula, majoriti kajian ini memberi tumpuan pada pengesanan maksimum dua kecacatan per wafer. Pendekatan ini tidak memenuhi keperluan industri untuk sistem pengesanan menyeluruh yang dapat mengenal pasti semua corak kecacatan yang ada pada wafer. Memandangkan bilangan kajian yang terhad mengenai pengelasan kecacatan jenis campuran dan keterbatasan kajian sedia ada, terdapat jurang penyelidikan yang signifikan dalam mengaplikasikan teknik pembelajaran mendalam, khususnya CNN, untuk mengelaskan corak kecacatan jenis campuran pada WBM dengan berkesan. Penyelidikan ini menggunakan iterasi terkini dari arkitektur YOLO, YOLOv8, untuk

mengklasifikasikan kecacatan jenis tunggal dan campuran melintasi dataset kompleks, Dataset MixedWM38, yang merangkumi 38 kelas pola kecacatan WBM. Dapatan menunjukkan bahawa model YOLOv8, khususnya variasi YOLOv8l, menunjukkan peningkatan dalam ketepatan klasifikasi dan kelajuan pemprosesan, mencapai hingga 98.4% ketepatan dan masa inferensi 0.7 milisaat. Keputusan ini melebihi prestasi model pembelajaran mendalam yang sedia ada. Penyelidikan masa depan harus memberi tumpuan pada peningkatan kekuatan model dan memastikan prestasi yang adil di semua kelas dengan menguatkuasakan data untuk kelas yang kurang diwakili dan menggunakan teknik resampling untuk mengimbangi distribusi kelas.

Kata Kunci: Pembuatan semikonduktor, pengelasan kecacatan peta bin wafer, pembelajaran mendalam, YOLOv8

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who provided guidance and support that contributed to the successful completion of this research project.

First, I am deeply thankful to my supervisor, Assoc. Prof. Ts. Dr. Aznul Qalid Md Sabri, whose expertise and understanding were instrumental in guiding the direction of this study. His guidance and support have been a great source of inspiration throughout this journey. Special thanks go to my peers at Faculty of Computer Science and Information Technology, Universiti Malaya, who provided both academic and personal support.

I am also thankful to the academic staff at the Faculty of Computer Science and Information Technology, Universiti Malaya, whose assistance in accessing necessary resources was invaluable. My gratitude also extends to my family and friends, who have provided unwavering support throughout my academic pursuits. Their constant encouragement has been a tremendous source of strength.

Thank you all for your invaluable contributions to this research project.

TABLE OF CONTENTS

Abstract	iii
Abstrak	v
Acknowledgements	vii
Table of Contents	viii
List of Figures	x
List of Tables	xi
List of Symbols and Abbreviations	xii
CHAPTER 1: INTRODUCTION.....	1
1.1 Research Background.....	1
1.2 Research Problem Statement.....	4
1.3 Research Questions	4
1.4 Research Objectives	4
1.5 Research Scope and Contribution	5
CHAPTER 2: LITERATURE REVIEW	6
2.1 Mixed-Type WBM Defect Classification	6
2.2 Difference between Wafer Surface Images and WBM	9
2.3 YOLO for Wafer Defect Detection and Classification	10
CHAPTER 3: METHODOLOGY.....	20
3.1 Flow Chart of Proposed Methodology	20
3.2 Data Collection.....	20
3.3 Data Pre-processing	26
3.3.1 Data Loading	26
3.3.2 Label Encoding.....	26
3.3.3 Data Validation and Correction	26
3.3.4 Data Splitting.....	28

3.3.5	Image Processing - Grayscale Conversion	29
3.3.6	Image Processing - Colour Encoding	30
3.3.7	Image Processing - Resizing	31
3.3.8	Directory Structure and Image Storage	31
3.4	Model Training	32
3.5	Evaluation of Model Performance	37
CHAPTER 4: RESULTS		38
CHAPTER 5: DISCUSSION		55
5.1	Framework and Configuration Overview	55
5.2	Comparative Analysis among YOLOv8 Variants	55
5.3	Comparative Analysis with Older YOLO Versions	58
5.4	Comparative Analysis with State-of-the-Art Models.....	59
5.5	Discrepancies in Model Performance Metrics	61
5.6	Performance Analysis for Specific Classes.....	63
CHAPTER 6: CONCLUSION.....		66
REFERENCES.....		68

LIST OF FIGURES

Figure 1.1: Single-type WBM defect patterns (Shinde et al., 2022).	3
Figure 1.2: Mixed-type WBM defect patterns (Wang, 2020).	3
Figure 2.1: SEM images of wafer surface defects (Cheon et al., 2019).	10
Figure 3.1: Flow chart of proposed methodology.	20
Figure 3.2: Numerical arrays of the first sample data.	21
Figure 3.3: Numerical arrays of the 23999th sample data.	212
Figure 3.4: Numerical arrays of the 11799th sample data.	212
Figure 3.5: Numerical arrays of the 17999th sample data.	213
Figure 3.6: Numerical arrays of the 6999th sample data.	213
Figure 3.7: Distribution of wafer defects.	24
Figure 3.8: Distribution of pixel values across all wafer images before cleaning.	27
Figure 3.9: Number of total images in training, validation and test sets.	28
Figure 3.10: Label counts in training, validation and test sets.	29
Figure 3.11: WBM images after grayscale conversion.	30
Figure 3.12: WBM images after colour encoding.	31
Figure 3.13: YOLOv8 network architecture diagram (Zhai et al., 2023).	33
Figure 3.14: Detailed module in the YOLOv8 network (Zhai et al., 2023).	34
Figure 5.1: Average performance metrics of YOLOv8 models across all classes.	57
Figure 5.2: Top-1 accuracy and inference time of YOLO models.	59
Figure 5.3: Model accuracy vs. FLOPS for YOLOv8 and state-of-the-art models.	61
Figure 5.4: Top-1 accuracy vs. average accuracy across all classes for YOLOv8 models.	63
Figure 5.5: Performance metrics of Random class across YOLOv8 models.	64
Figure 5.6: Performance metrics of Near-Full class across YOLOv8 models.	65

LIST OF TABLES

Table 2.1: Critical analysis table on mixed-type WBM defect classification.	15
Table 2.2: Critical analysis table on YOLO models for wafer defect detection.	17
Table 3.1: 38 classes of WBM defect patterns in MixedWM38 dataset.	24
Table 3.2: Hyperparameters for model training.	33
Table 3.3: Network parameters of YOLOv8n Pretrained Classify model (Ultralytics, 2024).	34
Table 3.4: Network parameters of YOLOv8s Pretrained Classify model (Ultralytics, 2024).	35
Table 3.5: Network parameters of YOLOv8m Pretrained Classify model (Ultralytics, 2024).	35
Table 3.6: Network parameters of YOLOv8l Pretrained Classify model (Ultralytics, 2024).	36
Table 3.7: Network parameters of YOLOv8x Pretrained Classify model (Ultralytics, 2024).	36
Table 3.8: Calculation formulae for model performance metric.	37
Table 4.1: Class-wise accuracy and precision of YOLOv8 models on test set.	38
Table 4.2: Class-wise recall and F1-score of YOLOv8 models on test set.	40
Table 4.3: Top-1 accuracy and inference time of YOLOv8 models on test set, and comparison with YOLO models in published literature.	42
Table 4.4: Comparison of YOLOv8 classification performance against existing models on MixedWM38 dataset.	42
Table 4.5: Learning curve and top-1 accuracy curve of YOLOv8 models on training set.	43
Table 4.6: Qualitative results on test set using YOLOv8l pretrained Classify model.	45
Table 5.1: Top-1 accuracy of YOLOv8 models on training and test sets.	57

LIST OF SYMBOLS AND ABBREVIATIONS

AE	:	Autoencoder
AP	:	Average Precision
BGR	:	Blue, Green, Red
C	:	Centre
CAE	:	Convolutional Autoencoder
CNN	:	Convolutional Neural Network
D	:	Donut
EL	:	Edge-Loc
ER	:	Edge-Ring
FN	:	False Negative
FP	:	False Positive
FPN	:	Feature Pyramid Network
FLOPS	:	Floating-Point Operations Per Second
GAN	:	Generative Adversarial Network
L	:	Loc
ML	:	Machine Learning
MAP	:	Mean Average Precision
NF	:	Near-Full
PAN	:	Path Aggregation Network
R	:	Random
RGRN	:	Randomized General Regression Network
SEM	:	Scanning Electron Microscopy
S	:	Scratch
SPP	:	Spatial Pyramid Pooling
SGD	:	Stochastic Gradient Descent
TN	:	True Negative
TP	:	True Positive
WBM	:	Wafer Bin Map
WBF	:	Weighted Box Fusion
YOLO	:	You Only Look Once
SEM	:	Scanning Electron Microscopy

CHAPTER 1: INTRODUCTION

1.1 Research Background

Semiconductor manufacturing can be divided into four primary stages namely wafer fabrication, wafer probe test, assembly, and final test. Wafer fabrication involves building integrated circuits on a silicon wafer through various complicated processes such as etching and photolithography. After fabrication, wafer probe test is conducted to evaluate basic functionality of each wafer. The results of a wafer probe test are represented by a wafer bin map (WBM). A silicon wafer is made up of basic units named dies. Depending on functionality, each die gets a bin colour on the WBM. For instance, defective dies are represented as yellow whereas qualified dies are represented as green on WBM as shown in Figure 1.1 (Shinde et al., 2022). The distribution of bin colours on WBM forms various defect patterns such as centre, donut, edge-loc, edge-ring, loc, near-full, scratch, and random. WBM defect patterns provide basis for root cause analysis to identify potential pitfalls in the manufacturing process leading to low product yields. These WBM defect patterns include single-type defect patterns, where there is only one type of defect on the wafer, and mixed-type defect patterns, where multiple types of defects occur simultaneously on the same wafer, such as a combination of donut and scratch defects as shown in Figure 1.2 (Wang, 2020).

Traditionally, wafer defect monitoring relies on manual evaluation and inspection by domain experts. This approach lacks efficiency and is often inadequate in terms of accuracy due to human error. Moreover, as semiconductor manufacturing becomes more advanced, the size of chips rapidly shrinks to nanometre range which makes manual inspection impractical. Therefore, semiconductor industry currently focuses on leveraging computer-aid methods specifically machine learning to automate wafer defect monitoring. Conventional machine learning (ML) has several limitations for image processing and pattern recognition. One of the greatest challenges is its reliance on

manual feature extraction, a process designed to transform raw data into distinctive properties. Additionally, conventional ML is incapable of handling large-scale, low-quality data. Due to the needs for manual feature extraction and noise filtering, conventional ML is computationally intensive which often leads to distortion of information, resulting in reduced accuracy of pattern recognition. In contrast, deep learning has the capability to autonomously identify high-level features when presented with basic input data, so there is no need for manual feature extraction (Tello et al., 2018). For this reason, deep learning is a promising tool for pattern recognition and classification. Many recent studies since 2018 have successfully demonstrated the application of deep learning for wafer defect classification. Generally, there are three types of deep learning algorithms applied for wafer defect classification including Convolutional Neural Network (CNN), Generative Adversarial Network (GAN) and Autoencoder (AE). Among these three networks, CNN is the most widely studied deep learning algorithm for wafer defect classification due to its robust algorithms and high classification accuracy (Batool et al., 2021).

CNNs are designed for processing data that has a grid structure, such as images. They consist of multiple layers: convolutional layers, pooling layers, and fully connected layers. In the convolutional layers, several filters are applied to the input to generate feature maps, which capture and summarize detected features. Pooling layers then simplify the feature maps by reducing their dimensionality but retaining critical information. Finally, the fully connected layers analyse these feature maps to produce predictions. CNNs are effective for image classification and have proven effective in tasks which require recognition and classification of visual patterns, such as defect patterns in semiconductor wafers (Yamashita et al., 2018).

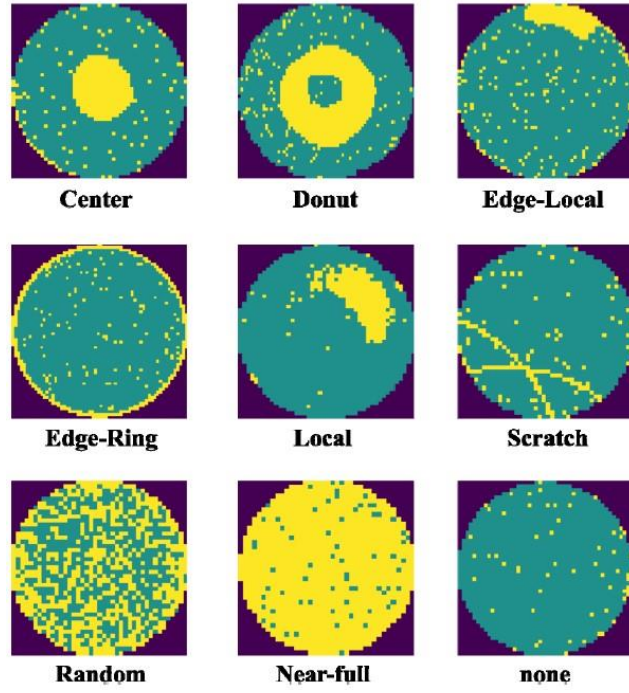


Figure 1.1: Single-type WBM defect patterns (Shinde et al., 2022).



Figure 1.2: Mixed-type WBM defect patterns (Wang, 2020).

1.2 Research Problem Statement

Despite the continuous surge in literatures since 2018 focusing on the application of CNN for wafer defect classification, majority of these studies primarily address single-type defect classification (Batoool et al., 2021). Single-type defect classification refers to classification of WBM that contains only one type of defect. However, in practical industry scenarios, a defective wafer often contains more than one type of defects. Accurately identifying all types of defects on a wafer is crucial for effective root cause analysis, which is necessary for enhancing manufacturing yield and reliability. Currently, only a limited number of studies have addressed the classification of mixed-type defects (Batoool et al., 2021; Yu et al., 2022; Nag et al., 2022). These studies typically utilize CNNs, either alone or in combination with other algorithms. Additionally, majority of these studies focused on detecting a maximum of two defects per wafer. This approach falls short of meeting the industry's needs for a comprehensive detection system that can identify all types of defect patterns present on a wafer. Given the limited number of research on mixed-type defect classification and the limitations of existing studies, a significant research gap remains in applying deep learning techniques, specifically CNNs, to classify mixed-type defect patterns on WBMs effectively.

1.3 Research Questions

The following two research questions are framed:

- i. What deep learning algorithm can be applied for classifying mixed-type WBM defect patterns?
- ii. How does the classification performance of the proposed model compare to existing models in the published literature?

1.4 Research Objectives

The following two research objectives are framed:

- i. To identify a deep learning algorithm for the classification of mixed-type WBM defect patterns.
- ii. To evaluate the classification performance of the proposed model against existing models in the published literature.

1.5 Research Scope and Contribution

The research scope focuses on identification and application of a deep learning algorithm for the classification of mixed-type WBM defect patterns, followed by evaluating the classification performance of the proposed model against existing models in the published literature. The contribution of this research work is to identify and apply a deep learning model which can accurately classify mixed-type WBM defect patterns. This research aims to enhance the automation of wafer defect monitoring to facilitate root cause analysis in semiconductor manufacturing, thus reducing the reliance on manual inspection by domain experts. Given the limited number of existing literatures focused on mixed-type WBM defect classification and the individual limitations of these studies, this research endeavours to address and fill the research gap. By improving WBM defect classification accuracy, this study could lead to increased operational efficiency, reduced costs, and higher product yields in semiconductor manufacturing processes.

CHAPTER 2: LITERATURE REVIEW

2.1 Mixed-Type WBM Defect Classification

Identifying multiple types of defects on a wafer is crucial for accurate defect classification and identifying underlying causes. If some defects are not identified, it can result in an incomplete analysis, adversely affecting enhancements in the wafer fabrication process and the overall product yield. This section provides an overview of the existing research on classifying mixed-type WBM defects (Batool et al., 2021).

Tello et al. (2018) proposed a system for classifying mixed-type defects through a combination of a randomized general regression network (RGRN) and a CNN. The process began by eliminating random defects through a spatial filter in the data preprocessing phase. Subsequently, the data was segmented into single-type and mixed-type defect categories utilizing a splitter informed by Information Gain theory. The RGRN was tasked with identifying single-type defects, while the CNN managed mixed-type defects. They assessed their system using datasets that were both real and synthetic, including seven categories of defects—three single-type and four mixed-type. The splitter was crucial in achieving a maximum accuracy of 95%.

Kyeong and Kim (2018) presented a CNN-only approach for classifying mixed-type defects, using four separate CNN models for four classes of single-type defects. They found that using individual models for each defect class outperformed the use of a single model for all defect classes. Their system was evaluated on 16 defect classes as combinations of the four single-type defects, using synthetic data for training and a small set of real data for testing. However, their system could not count multiple occurrences of the same type of defect on a wafer. While their ensemble of models facilitated the integration of new defect types and addressed class imbalance, it required significant resources.

Devika and George (2019) optimized a single CNN trained on four classes of single-type defect. The model was tested on 8 defect classes as combinations of the four single-type defects. They augmented their dataset by drawing patterns in paint, which is a simplistic approach for such a sensitive task. Their model's performance was not compared against any benchmarks, limiting the evaluation of its effectiveness.

Kong and Ni (2019) developed a multi-step system for classifying wafers with overlapping and non-overlapping patterns. Initially, a binary CNN classified wafers based on the presence or absence of overlapping patterns. For wafers with non-overlapping mixed-type patterns, a seed filling algorithm was used for segmentation, followed by classification through a CNN. Overlapping patterns were addressed using a template matching method. In a subsequent enhancement in 2020, they integrated UNet for defect boundary segmentation alongside CNN for classification, resulting in improved accuracy on a real dataset with seven classes of defect patterns (Kong & Ni, 2020).

Byun and Baek (2020) introduced a variation in the standard CNN architecture by initializing CNN weights with a convolutional autoencoder (CAE). They trained their model on single-type WBM defect patterns and evaluated it on mixed-type defect patterns, classifying defects according to probability thresholds. Utilizing the WM-811K dataset, they excluded the non-pattern class and developed five new classes for mixed-type defects. While the model showed strong performance for single-type defects, it struggled with accurately classifying mixed-type defects.

Hyun and Kim (2020) proposed a memory-augmented CNN equipped with triplet loss for handling highly imbalanced dataset containing mixed-type WBM defect patterns. The CNN, employing triplet loss, served as an embedding tool to transform high-dimensional data into a more manageable lower dimension. To tackle class imbalance, a key-value memory module was introduced, which allocated equal memory resources to each class.

Their system's effectiveness was tested using a synthetic dataset comprising 16 classes, which included one non-pattern, four single-type, and eleven mixed-type defect classes

Wang et al. (2020) proposed a CNN equipped with deformable convolutional module and one-hot encoding which promotes selective sampling and extracting high-quality features from mixed-type wafer defect datasets. Despite its good classification performance, the complexity of the proposed model could pose practical challenges. They also released a public dataset called "MixedWM38," containing 38 classes of WBM defect patterns (9 single-type and 29 mixed-type defect classes).

Yu et al. (2022) proposed a method to reduce the complexity of deep models without compromising the representation of wafer features by adjusting the dense block structure in the PeleeNet network. They introduced a lightweight network called WM-PeleeNet, based on the PeleeNet module. Tested on MixedWM38 dataset, the average accuracy of WM-PeleeNet is 97.5%, significantly higher than the 93.5% accuracy achieved by the dataset sharer Wang et al. (2020).

Nag et al. (2022) presented WaferSegClassNet (WSCN), a novel network based on encoder-decoder CNN architecture designed for the simultaneous classification and segmentation of mixed-type wafer defects. Evaluated on the MixedWM38 dataset, WSCN achieved an average classification accuracy of 98.2%. WSCN is the first model in wafer defect analysis that performs both segmentation and classification simultaneously.

This section reviews the literature on mixed-type WBM defect classification. Various studies have tackled the issue of identifying mixed-type defects, often employing CNNs in different configurations. These include standalone CNNs, ensembles of CNNs, and CNNs combined with other algorithms. Both real and synthetic data have been used for model training and validation. Synthetic data is commonly used for the study of mixed-

type defects because real datasets often lack wafers with mixed-type defects. Due to the relative rarity of mixed-type defects compared to single-type defects in real datasets, many researchers have generated WBM containing mixed-type defects for model evaluation. Balanced training sets were often created, but less attention was given to addressing data imbalance. Additionally, most studies focused on detecting a maximum of two defects per wafer, which is not a comprehensive approach. For a model to be practical and effective in the real-world industry, it should be capable of identifying all types of defect patterns on a wafer. Table 2.1 summarizes the findings of the literature reviewed in this section.

2.2 Difference between Wafer Surface Images and WBM

In semiconductor manufacturing, the analysis and classification of wafer defects are dependent on the data gathered during production and testing. Two types of data are often gathered in this process: wafer surface images and wafer bin map (WBM) data.

Wafer surface images are used to identify physical defects on the wafer surface, such as cracks, contamination, and structural imperfections. Wafer surface images are captured using Scanning Electron Microscopy (SEM) or optical microscopes, providing a detailed view of the wafer's surface at a microscopic level. SEM images are useful for examining surface topography, detecting structural defects, and analysing material composition at high resolution, which are vital for understanding the potential impact on circuit functionality. In contrast, optical microscopy serves as a faster and less costly method for initial surface inspection at low resolution, helping to quickly identify larger and more visible defects (Cheon et al., 2019).

On the other hand, WBM data provides a different insight. These maps are visual representations of the outcomes from various wafer tests, depicting the performance of each die on the wafer. Each die is assigned a colour based on its test results, indicating whether it passed, failed, or how well it performed against specific benchmarks. This data

is used for quality control and yield analysis, allowing for a quick assessment of overall wafer health and the identification of patterns in die performance. These patterns can be crucial for diagnosing process issues and strategizing improvements in manufacturing processes (Batoool et al., 2021).

Together, wafer surface images and WBM data offer complementary perspectives that are essential for a comprehensive approach to defect classification in semiconductor manufacturing. While surface images provide a micro-level view of individual defects, WBMs give a macro-overview of die performance across the entire wafer, combining to form a complete picture that helps improve root cause analysis, and ultimately increase the yield of semiconductor products.

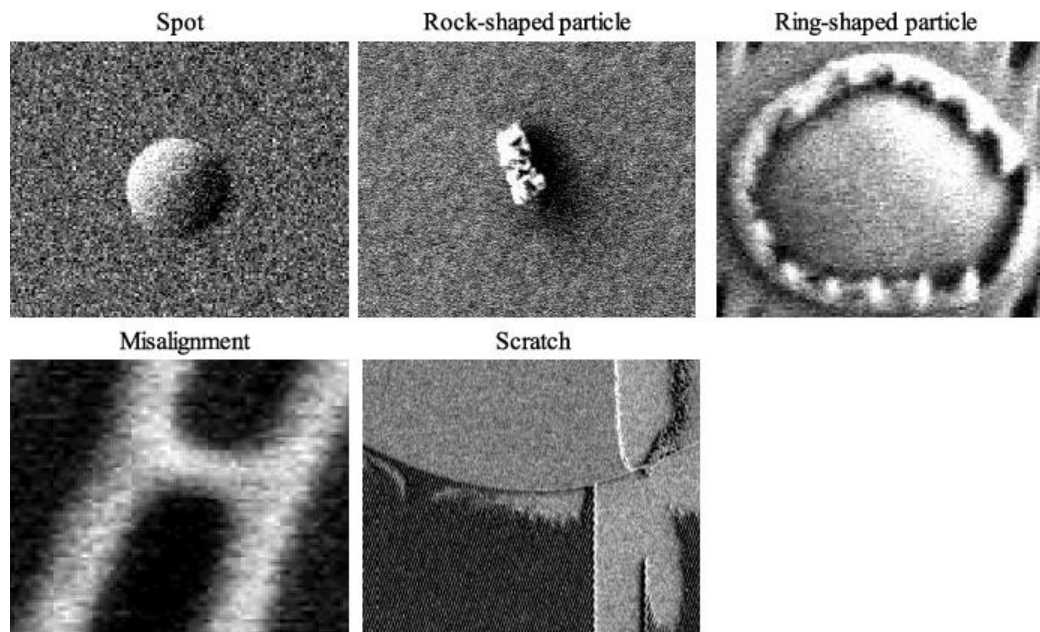


Figure 2.1: SEM images of wafer surface defects (Cheon et al., 2019).

2.3 YOLO for Wafer Defect Detection and Classification

The YOLO (You Only Look Once) object detection algorithm is a groundbreaking approach in the field of computer vision (Redmon et al., 2016). Developed as a type of CNN, YOLO is used to detect and classify multiple objects in images with remarkable speed and accuracy. Since its inception, YOLO has undergone several iterations, each enhancing its architecture to improve both accuracy and processing speed. This algorithm

is significant in industrial applications where real-time detection is crucial. In the context of semiconductor manufacturing, YOLO has been adapted for wafer defect detection and classification. Its ability to process images quickly and efficiently makes it an ideal choice for identifying various defects on wafer surfaces, such as spots, cracks, and micropipes. The continuous improvements in YOLO models enabled them to handle the high-resolution images in semiconductor inspections, ensuring reliable defect detection and classification.

Cao et al. (2020) proposed a wafer surface defect detection method using an improved YOLOv3 network. Through the addition of Mish activation function and the Spatial Pyramid Pooling (SPP) module, the modified YOLOv3 model achieved an average detection accuracy of 85.1%, which is approximately 13% higher than the original YOLOv3 model. The detection speed is about 50% faster than the Faster-RCNN model, meeting the requirement for real-time defect detection.

Chen et al. (2020) applied deep learning methods for detecting die particle defects. They utilized a Generative Adversarial Network (GAN) to generate pseudo defective images from real defective images, thereby enhancing the diversity of training set. For defect detection, they used YOLOv3, which determined the presence of defects through the bounding boxes it predicts. The combined use of GAN and YOLOv3 eliminated the need for a large collection of defective samples, making it advantageous for handling various defect patterns.

Li et al. (2021) applied the YOLOv4 target detection algorithm to silicon wafer crack defect detection and managed to achieve a 98.23% detection accuracy on the wafer crack detection dataset. The algorithm effectively handles varying crack sizes and lengths, ensuring accurate detection without over-identifying line marks, demonstrating robust performance for industrial silicon wafer crack detection.

Shinde et al. (2022) evaluated the YOLO architecture for its ability to locate and classify WBM defects. Experiments on WM811K wafer map dataset demonstrated that YOLOv3 and YOLOv4 variants achieved over 94% classification accuracy in real-time. In comparison, other architectures like ResNet50 and DenseNet121 yielded accuracies of 89% and 92% respectively without localization capabilities. The study highlights the effectiveness of YOLO in both locating and classifying defects on WBM.

Zhou et al. (2022) improved the YOLOv5 model for the detection of micropipe defects on silicon wafers. This model enhances high-resolution feature extraction by adding a detection branch in the neck and head blocks, while integrating a CBAM and a dual-attention module to capture spatial and channel attention. The adjustment effectively increased the mean Average Precision (mAP) by 1.89% as compared to the original YOLOv5 model.

Bhardwaj et al. (2023) implemented YOLOv8 to detect and classify three types of defects in wafer surface images: spot, crack, and no defect. The model achieved a mAP of 94.2%, with 93% precision and 92.7% recall. Future research will focus on training the algorithms on extensive datasets to improve defect detection accuracy.

Dehaerne et al. (2023) optimized the YOLOv7 model for defect detection in SEM images by adjusting hyperparameters and combining model predictions. Adjusting each hyperparameter improved Average Precision (AP) for specific defect classes. Combining predictions from models with the best APs for different classes, using the Weighted Box Fusion (WBF) method, increased mAP by 10% compared to a single YOLOv7 model with default settings. Future work could further improve results with advanced hyperparameter optimization techniques.

Kumar et al. (2023) utilized YOLOv5 models to detect wafer defects in wafer surface images captured by SEM or optical microscopes, focusing on two types of defects: cracks

and particles. The YOLOv5 model achieved an 89% mAP, surpassing traditional wafer defect detection methods. Future research will aim to address a broader range of defects in SEM images.

Choi et al. (2024) employed YOLOv8 for analysing SEM images from wafer inspection machines for defect localization and classification. Leveraging its balance between accuracy and inference speed, YOLOv8 model effectively predicted six types of defects, achieving a mAP of 0.789 on unseen test data collected from five different wafer fabrication facilities with varying image qualities.

Yang et al. (2024) proposed CTM-IYOLOv5, a wafer surface defect detection method based on YOLOv5. An inspection platform captured real-time wafer surface images. To address false detections caused by multiple inter-grain channels, clustering-template matching was used to segment the grains within the field of view. GhostNet was introduced into the network structure of YOLOv5 to achieve a lightweight model. The proposed method achieved an accuracy of up to 99% with improved inspection efficiency, making it ideal for real-time defect detection.

This review illustrates that most of the research utilizing YOLO models for wafer defect detection has focused predominantly on wafer surface images, leveraging their capability to identify and classify defects in high-resolution wafer surface images. Notably, only one study has explored the application of YOLO models on WBM, and this was limited to single-type defects encompassing nine classes (Shinde et al., 2022). Given the effectiveness of YOLO in handling complex image data, there is a significant opportunity to extend its application to mixed-type defect classification in WBM, which can include up to thirty-eight distinct classes. Such an expansion could harness YOLO's robust detection capabilities to extract insights from WBM defect patterns, enhancing both the accuracy and processing speed of WBM defect classification. This approach opens new avenues for research in the monitoring of semiconductor product quality,

where identification of multiple defect types on a wafer is crucial. Table 2.2 summarizes the findings of the literature reviewed in this section.

Table 2.1: Critical analysis table on mixed-type WBM defect classification.

Citations	Theoretical / Conceptual Framework	Dataset	Model / Methodology	Results	Implications for Future Research
(Nag et al., 2022)	Proposed WaferSegClassNet, a model based on encoder-decoder CNN architecture for simultaneous classification and segmentation of mixed-type wafer defects.	MixedWM38 dataset containing 38 classes of defect patterns (9 classes of single-type defect, 29 classes of mixed-type defect)	Encoder-decoder CNN	98.2% accuracy; 98.08% precision; 98.18% recall	Future research will expand this defect analysis technique to identify defects in the context of civil construction.
(Yu et al., 2022)	Proposed WM-PeleeNet, a lightweight CNN based on PeeleNet model for classification of mixed-type wafer defects. Reduce the model complexity by adjusting the dense block structure in the PeeleNet network.	MixedWM38 dataset containing 38 classes of defect patterns (9 classes of single-type defect, 29 classes of mixed-type defect)	CNN based on PeeleNet model	97.5% accuracy	The model presented in this paper struggled with misclassifications of Loc and Edge-Loc patterns. Future research will aim to achieve high-precision classification of Loc and Edge-Loc patterns.
(Wang et al., 2020)	Proposed a customized CNN with deformable convolutional module and one-hot encoding which promotes selective sampling and extracting high-quality features from mixed-type wafer defects.	MixedWM38 dataset containing 38 classes of defect patterns (9 classes of single-type defect, 29 classes of mixed-type defect)	CNN with deformable convolutional module and one-hot encoding	93.2% accuracy; 94% precision; 95% recall	Future research will focus on defect root cause detection and modelling wafer defects.
(Hyun & Kim, 2020)	Proposed a memory-augmented CNN with triplet loss for handling imbalanced defect data. Triplet loss acted as an embedding function to map high-dimensional data to a lower dimension. A key-value memory module managed class imbalance by allocating equal memory for each class.	Simulated dataset containing 16 classes of WBM defect patterns (5 classes of single-type defect, 11 classes of mixed-type defect)	Memory-augmented CNN with triplet loss	88.8% accuracy; 93.1% precision; 75.6% recall	Future research will study more classes of defect patterns. The increase in the number of defect classes may deteriorate the model performance.

(Byun & Baek, 2020)	Proposed a customized CNN by initializing CNN weights with a convolutional autoencoder (CAE). Trained on single-type defect data and tested on mixed-type defect data, the model distinguished defect categories based on probability and a threshold value.	WM811K dataset containing 8 classes of single-type defect patterns. Generated additional 5 classes of mixed-type defect patterns.	CAE (initialization of weights), CNN	The model performed well for single-type defects (92.9% accuracy) but had difficulty with mixed-type defects (54.8% accuracy).	This work assumed a maximum of two types of defect patterns per wafer. Future research will study for more mixed-type patterns.
(Kong & Ni, 2020)	Proposed a classification system by combining UNet for defect boundary segmentation and CNN for classification.	Real-world dataset containing 7 classes of single-type WBM defect patterns. Generated additional 8 classes of mixed-type defect patterns.	UNet, CNN	90.10% accuracy	Future research will study more classes of defect patterns.
(Kong & Ni, 2019)	Proposed a multistep system for classifying wafers with overlapping and non-overlapping mixed-type defect patterns. Non-overlapping patterns were segmented using a seed filling algorithm and then classified by a CNN. Overlapping patterns were classified using a template matching technique.	2 real-world datasets containing 12 and 9 classes of single-type WBM defect patterns respectively. Generated additional 5 classes of mixed-type defect patterns for each dataset.	CNN, seed filling segmentation, template matching method	91.2% and 84.3% accuracy (on two real-world datasets)	The newly generated defect patterns lack variety due to the fixed, regular changes in the size and position of the defect pattern generated by morphological transformation.
(Devika & George, 2019)	Employed a single CNN trained on four basic defect types to detect combinations of these basic types. They augmented their dataset by drawing patterns in paint, which is a simplistic approach for such a sensitive task.	Real and simulated dataset containing 8 classes of WBM defect patterns (4 classes of single-type defect, 4 classes of mixed-type defect)	CNN	84% accuracy in mixed-type defect classification.	Current work generated more defect patterns by drawing patterns in paint. This approach is not reliable for scientific investigation.

(Kyeong & Kim, 2018)	Proposed a CNN-ensemble approach for classifying mixed-type defects, using four separate CNN models for the four basic defect types. Four models were applied on each wafer defect data.	Real and simulated dataset containing 16 classes of WBM defect patterns (5 classes of single-type defect, 11 classes of mixed-type defect)	CNN ensemble	97.4% accuracy; 98.2% precision; 97.4% recall	Future research will study more classes of defect patterns.
(Tello et al., 2018)	Proposed a framework utilizing a randomized general regression network (RGRN) for single-type defect classification and CNN for mixed-type defect classification.	Real-world dataset containing 7 classes of single-type WBM defect patterns. Generated additional classes of mixed-type defect patterns.	RGRN, CNN	86.17% accuracy	Future research will evaluate the model using datasets that represent current realistic wafer volumes while also managing the increased computational complexity.

Table 2.2: Critical analysis table on YOLO models for wafer defect detection and classification.

Citations	Theoretical / Conceptual Framework	Dataset	Model / Methodology	Results	Implications for Future Research
(Yang et al., 2024)	Improved YOLOv5 by introducing GhostNet into the original network structure to achieve a lightweight model which enabled real-time defect detection on wafer surface images.	Wafer surface images taken by inspection platform, containing 3 types of defects – stain, collapse, and cutoff	Improved YOLOv5	99.5% mAP; 99.2% precision	Future research will study more types of defects in wafer surface images.
(Choi et al., 2024)	Employed YOLOv8 to locate and classify wafer defects for analysing Scanning Electron Microscope (SEM) images from wafer inspection machines.	SEM images from wafer inspection machines, containing 6 types of defects	YOLOv8	78.9% mAP	Future research will study more types of defects in SEM images.

(Kumar et al., 2023)	Employed YOLOv5 to detect wafer defects in SEM images. The study focused on 2 types of defects - cracks and particles.	Wafer surface images taken by SEM or optical microscope, containing 2 types of defects	YOLOv5	89% mAP; 81.6% precision, 80.8% recall	Future research will study more types of defects in SEM images.
(Dehaerne et al., 2023)	Optimized the YOLOv7 model architecture through hyperparameter tuning and combining predictions from different models.	SEM image dataset containing 5 types of defects – line collapse, bridge, microbridge, gap, and probable gap	Optimized YOLOv7, ensemble models	86.8% mAP	Future work could further improve results with advanced hyperparameter optimization techniques.
(Bhardwaj et al., 2023)	Implemented YOLOv8 for the detection and classification of defects in wafer surface images.	Wafer surface images, containing 3 types of defects – spot, crack, and no defect	YOLOv8	94.2% mAP; 93% precision; 92.7% recall	Future research will train the algorithms on extensive datasets to improve defect detection.
(Zhou et al., 2022)	Improved the YOLOv5 by adding a detection branch in the neck and head block to capture high-resolution features to enhance detection of micropipe defects on wafer surfaces.	Wafer surface images containing micropipe defects	Improved YOLOv5	79.8% mAP; 81.36% precision; 96.5% recall; 88.28% F1-score	Future work will focus on reducing the model's size and expanding the dataset to include a wider range of defects.
(Shinde et al., 2022)	Implemented YOLOv3 and YOLOv4 models to localize and classify wafer defects on wafer bin map dataset.	WM811K dataset containing 9 classes of single-type WBM defect patterns.	YOLOv3, YOLOv4	95.7% accuracy; 96% precision; 0.92 F-score	Future research will study defect localization and classification using unsupervised or semi-supervised methods.
(Li et al., 2021)	Implemented YOLOv4 model to detect cracks on silicon wafer. It effectively adapted to various sizes and lengths of cracks, ensuring accurate detection without over-identifying line marks.	Wafer surface images containing crack defects	YOLOv4	98.23% mAP; 96% precision; 99% recall; 0.97 F1-score	Future work will train the algorithm on larger dataset to include a wider range of defects.

(Chen et al., 2020)	Employed generative adversarial network (GAN) to generate pseudo defective images from the real defective images, enhancing the diversity of training set. YOLOv3 was employed to detect particle defects on wafer surfaces.	Wafer surface images containing particle defects	GAN, YOLOv3	88.72% mAP	Future research will study the potential of applying the proposed method on various defect patterns.
(Cao et al., 2020)	Improved the YOLOv3 model to improve the detection accuracy of small targets on wafer surfaces by introducing Mish activation function and Spatial Pyramid Pooling (SPP) module into the original network.	Wafer surface defect images	Improved YOLOv3	85.1% mAP	Future research will focus on applying the proposed method on real-time detection of wafer surface defects.

CHAPTER 3: METHODOLOGY

3.1 Flow Chart of Proposed Methodology

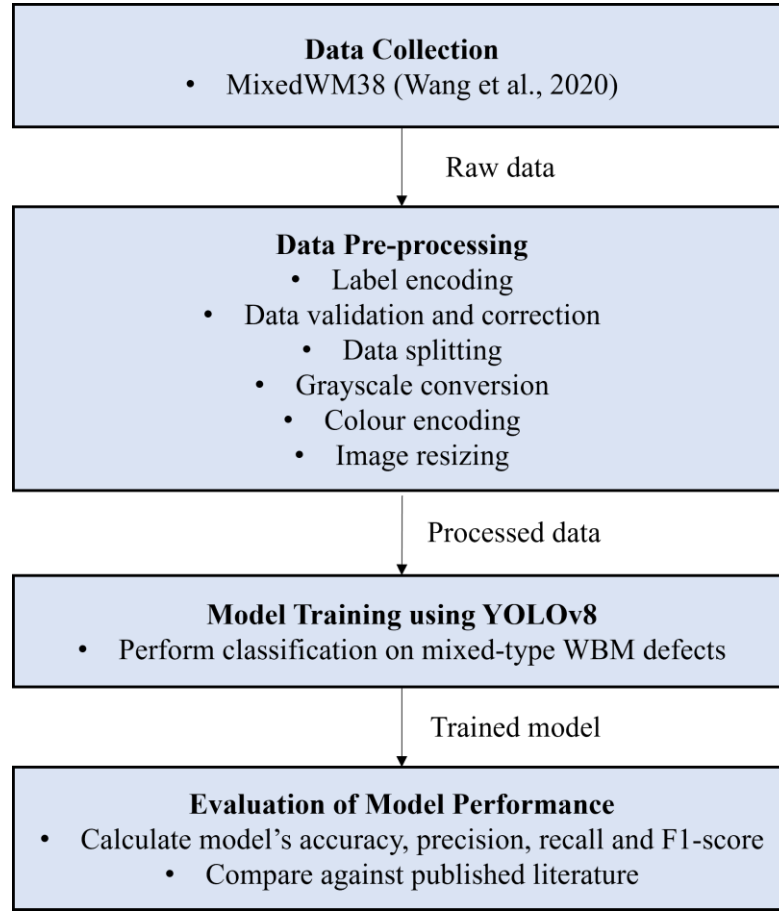


Figure 3.1: Flow chart of proposed methodology.

3.2 Data Collection

The proposed research is a quantitative research using secondary data. The WBM dataset named MixedWM38 is used in the project. MixedWM38 is prepared by Wang et al. (2020). The authors collected 38 types of WBM defect patterns from real-world fabrication. To keep the balance between various classes of defect patterns, the authors employed the generative adversarial networks (GAN) to generate samples for those defect classes with insufficient number of samples. The dataset is available on the authors' GitHub repository associated with the research article (Wang, 2020). It contains 38,015 wafer map images divided into 1 normal pattern (no defect), 8 single-type defect classes and 29 mixed-type defects classes. The WBM images are of 52×52 dimensions, each

represented as a numerical array within a 52×52 grid. Each pixel in these binary maps represents one of three states: non-wafer area (0), passing die (1) or failing die (2). On the other hand, the defect class label is provided as a one-hot encoded array in NumPy format.

[illegible]

Figure 3.2: Numerical arrays of the first sample data.

[illegible]

Figure 3.3: Numerical arrays of the 23999th sample data.

[illegible]

Figure 3.4: Numerical arrays of the 11799th sample data.

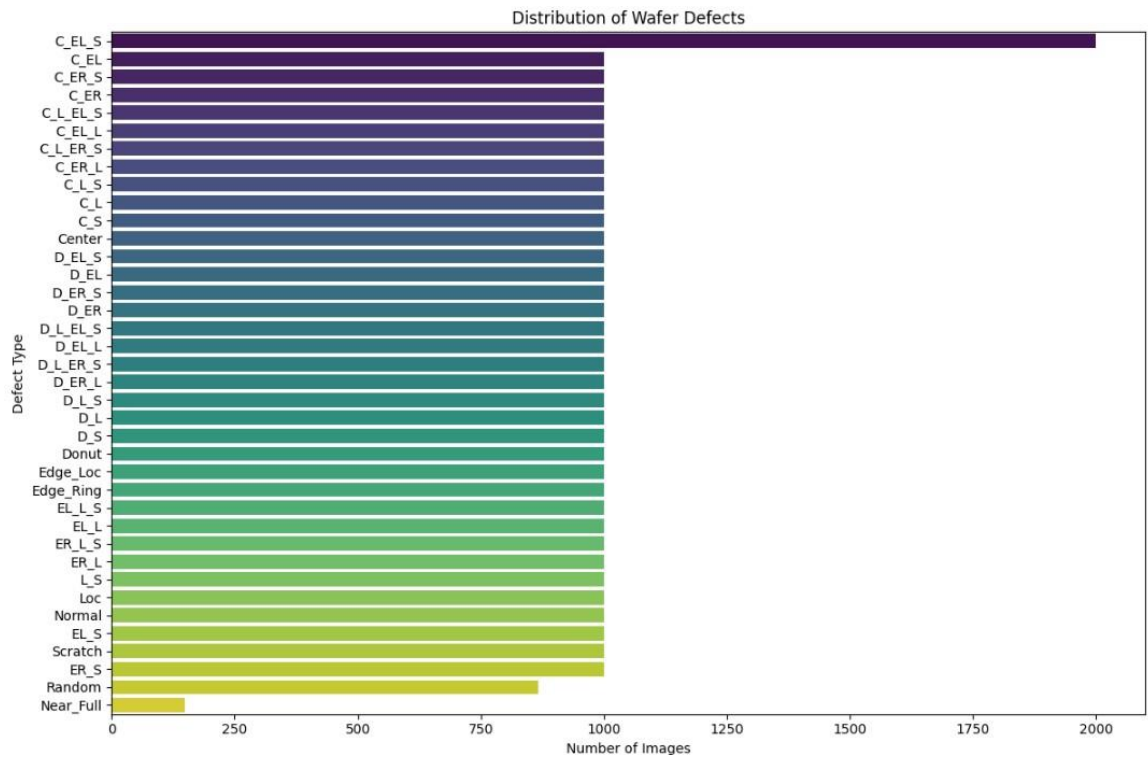


Figure 3.7: Distribution of wafer defects.

Table 3.1: 38 classes of WBM defect patterns in MixedWM38 dataset.

Patten Type	No.	Pattern Name	Pattern Label	Number of Images
Single-type defect	1	Normal	[0 0 0 0 0 0 0 0]	1000
	2	Centre (C)	[1 0 0 0 0 0 0 0]	1000
	3	Donut (D)	[0 1 0 0 0 0 0 0]	1000
	4	Edge-Loc (EL)	[0 0 1 0 0 0 0 0]	1000
	5	Edge-Ring (ER)	[0 0 0 1 0 0 0 0]	1000
	6	Loc (L)	[0 0 0 0 1 0 0 0]	1000
	7	Near-Full (NF)	[0 0 0 0 0 1 0 0]	149
	8	Scratch (S)	[0 0 0 0 0 0 1 0]	1000
	9	Random (R)	[0 0 0 0 0 0 0 1]	866
2 mixed-type defects	10	C+EL	[1 0 1 0 0 0 0 0]	1000
	11	C+ER	[1 0 0 1 0 0 0 0]	1000
	12	C+L	[1 0 0 0 1 0 0 0]	1000

	13	C+S	[1 0 0 0 0 0 1 0]	1000
	14	D+EL	[0 1 1 0 0 0 0 0]	1000
	15	D+L	[0 1 0 0 1 0 0 0]	1000
	16	ER+L	[0 0 0 1 1 0 0 0]	1000
	17	EL+S	[0 0 1 0 0 0 1 0]	1000
	18	ER+S	[0 0 0 1 0 0 1 0]	1000
	19	L+S	[0 0 0 0 1 0 1 0]	1000
	20	D+ER	[0 1 0 1 0 0 0 0]	1000
	21	D+S	[0 1 0 0 0 0 1 0]	1000
	22	EL+L	[0 0 1 0 1 0 0 0]	1000
3 mixed-type defects	23	C+EL+L	[1 0 1 0 1 0 0 0]	1000
	24	C+EL+S	[1 0 1 0 0 0 1 0]	2000
	25	C+ER+L	[1 0 0 1 1 0 0 0]	1000
	26	C+ER+S	[1 0 0 1 0 0 1 0]	1000
	27	C+L+S	[1 0 0 0 1 0 1 0]	1000
	28	D+EL+L	[0 1 1 0 1 0 0 0]	1000
	29	D+EL+S	[0 1 1 0 0 0 1 0]	1000
	30	D+L+S	[0 1 0 0 1 0 1 0]	1000
	31	D+ER+L	[0 1 0 1 1 0 0 0]	1000
	32	D+ER+S	[0 1 0 1 0 0 1 0]	1000
	33	EL+L+S	[0 0 1 0 1 0 1 0]	1000
	34	ER+L+S	[0 0 0 1 1 0 1 0]	1000
4 mixed-type defects	35	C+L+EL+S	[1 0 1 0 1 0 1 0]	1000
	36	C+L+ER+S	[1 0 0 1 1 0 1 0]	1000
	37	D+L+EL+S	[0 1 1 0 1 0 1 0]	1000
	38	D+L+ER+S	[0 1 0 1 1 0 1 0]	1000

3.3 Data Pre-processing

The data pre-processing stage is crucial for preparing the WBM dataset for effective training and evaluation of the machine learning models. This section outlines the steps taken to transform raw wafer images into a structured format suitable for analysis.

3.3.1 Data Loading

The process begins by loading the dataset, which comprises 38,015 wafer images, each represented as a numerical array within a 52×52 grid. These images are paired with corresponding labels that categorize the wafers into one of 38 defect classes. This dataset is loaded from a “.npz” file, a compressed NumPy format that allows for efficient storage and retrieval of large array-based datasets.

3.3.2 Label Encoding

Once loaded, the dataset's labels are decoded using a pre-defined mapping stored in a YAML file. The YAML file provides a structured way to map arrays of numeric labels to human-readable strings, which describe various types of wafer defects. Each entry in the YAML file corresponds to a specific type of defect or combination of defects. For instance, the label `[0, 1, 0, 0, 0, 0, 0, 0]` in the array format uniquely identifies the defect type "Donut". Similarly, a more complex defect pattern like `[1, 0, 1, 0, 1, 0, 0, 0]`, which represents a combination of "Centre", "Edge-Loc", and "Loc" defects, is labelled as "C+EL+L". When the dataset is loaded, each wafer's label array is compared against the mappings in the YAML file. The corresponding string descriptor is then assigned to the wafer, replacing the numeric array. This translation facilitates easier data handling and analysis since the labels are now meaningful descriptions of the defects.

3.3.3 Data Validation and Correction

In the data pre-processing phase for WBM dataset, an essential task is validating the integrity of the wafer map images. Each pixel in these binary maps should represent one

of three states: non-wafer area (0), passing die (1) or failing die (2). This classification ensures that each pixel accurately reflects the condition of die. However, discrepancies were identified in the dataset where 105 wafer maps contained an additional erroneous pixel value of 3, which does not conform to the expected values. All these erroneous maps were categorized under the "Near-Full" type defect label.

To address this issue, a validation script was executed to inspect the pixel values of each wafer map image. The script identified and listed wafer maps with more than the standard three distinct pixel values, revealing that some maps contained the value 3, which is outside the specified range of valid pixel values. This anomaly suggests a data entry error or a misclassification at the data collection or digitization stage. Once identified, these out-of-range pixel values were corrected by replacing them with the value 2, which corresponds to the 'failing die' condition. This substitution assumed that the erroneous value was intended to represent an extreme condition, closely aligning with the 'failing' state. This validation and correction step is vital for the subsequent phases of machine learning modelling, where accurate and consistent data input is crucial for training effective models.

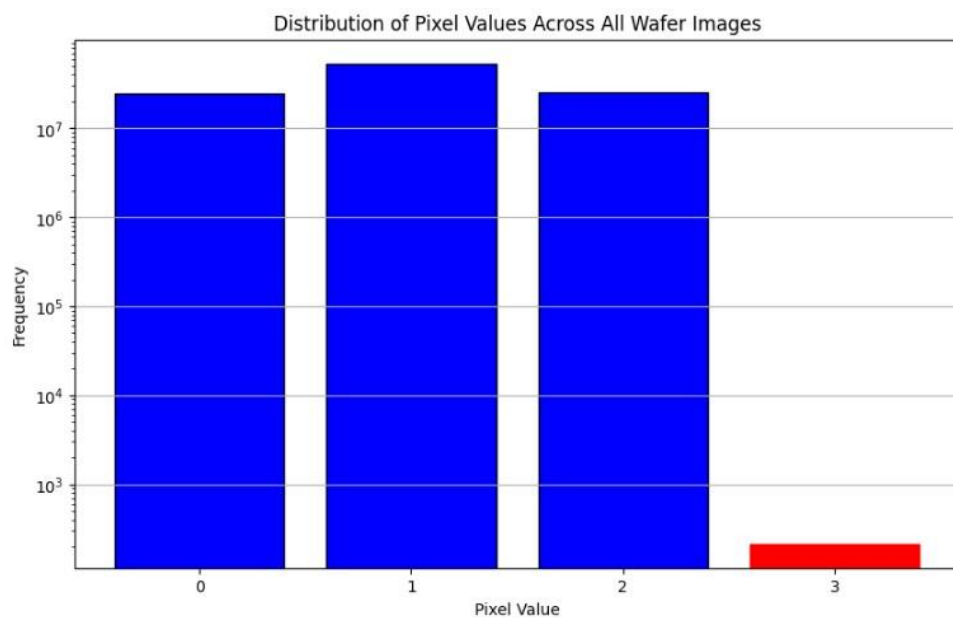


Figure 3.8: Distribution of pixel values across all wafer images before cleaning.

3.3.4 Data Splitting

The dataset is split into three subsets: training, validation, and testing. Specifically, 30% of the entire dataset is reserved for the test set, which provides an unbiased evaluation of the model after training and validation. Of the remaining 70% of the dataset, 20% (which mathematically represents 14% of the total dataset) is set aside for validation. This validation set plays a crucial role in tuning the model's parameters, helping to prevent overfitting and to adjust for biases or variance issues that may not be apparent from training alone. The rest, which accounts for 56% of the total dataset, is used as the training data. This majority share allows the model to learn from a diverse set of examples, equipping it with the ability to generalize well in real-world operations. In other words, the training set contains 21288 images whereas the validation set contains 5322 images and the test set contains 11405 images.

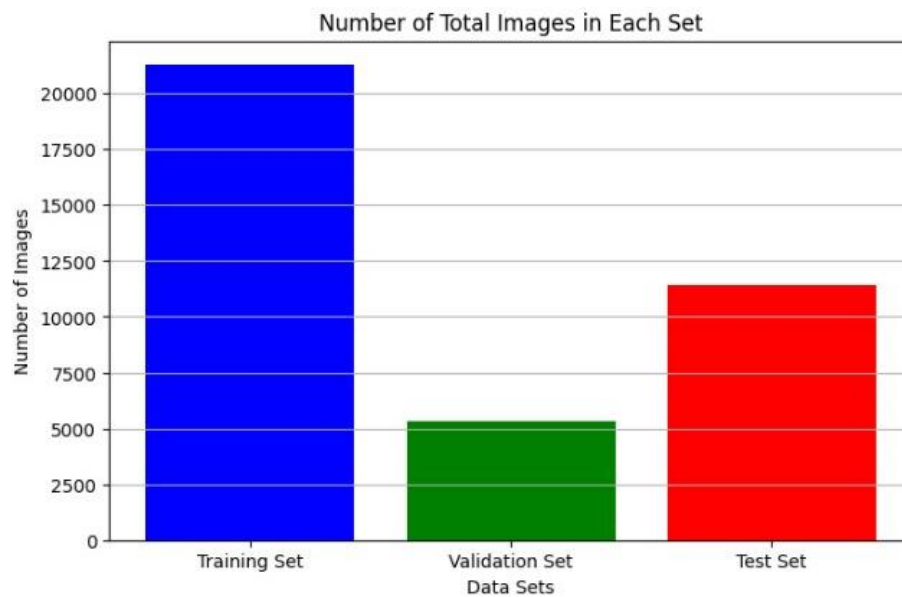


Figure 3.9: Number of total images in training, validation and test sets.

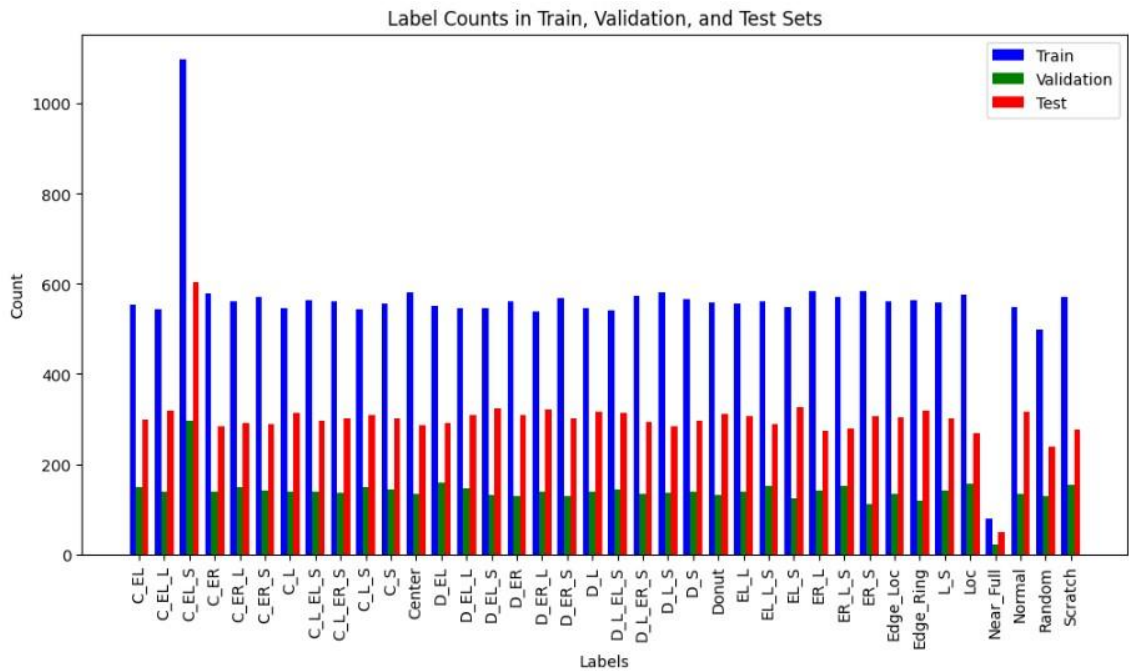


Figure 3.10: Label counts in training, validation and test sets.

3.3.5 Image Processing - Grayscale Conversion

The raw wafer map images, each originally presented as a numerical array, undergo grayscale conversion to facilitate easier visualization and more effective analysis. Each pixel in these arrays represents different conditions of the wafer surface, with values specifically designated as 0 for non-wafer areas, 1 for good areas, and 2 for defective areas. During the conversion process, these values are mapped to grayscale intensities: 0 remains black, signifying non-wafer areas; 1 is transformed to 128, a medium grey indicating good wafer condition; and 2 is converted to 255, white, highlighting defective regions. This grayscale mapping enhances the visual contrast among different wafer states, making it simpler to distinguish and evaluate them.

The transformed data is then cast into an 8-bit unsigned integer format, the standard for image processing. This format is compatible with a wide range of image processing tools and libraries, ensuring that the grayscale images can be easily integrated into various analysis and machine learning pipelines. The result is a set of grayscale images that not

only convey the condition of the wafers intuitively but also support subsequent analytical tasks by presenting the data in a universally accessible format.

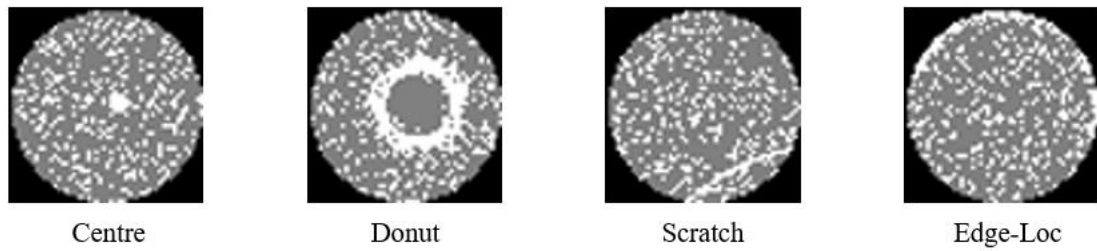


Figure 3.11: WBM images after grayscale conversion.

3.3.6 Image Processing - Colour Encoding

To further enhance visual differentiation and analytical clarity, the grayscale images are transformed into colour-coded images through a process of colour encoding. This step involves mapping the grayscale values, which represent different wafer conditions, to specific colours using the BGR (Blue, Green, Red) colour model. In this step, a NumPy array that represents a wafer map in grayscale—where the values 0, 128, and 255 indicate non-wafer areas, good wafer areas, and defective wafer areas respectively—is converted to display more intuitive colours. Non-wafer areas (0 in grayscale) are typically left unchanged to maintain a clear visual baseline. Good wafer areas (128 in grayscale) are encoded with an aqua colour (BGR: 25, 102, 255). This choice of colour not only stands out visually against the darker and lighter tones but also symbolizes stability and reliability. Defective wafer areas (255 in grayscale) are marked with a pumpkin colour (BGR: 255, 255, 0). This bright, alerting colour highlights areas of concern, drawing immediate attention to potential issues. This colour substitution process not only enhances the visual appeal of the maps but significantly improves the ease with which different conditions can be distinguished.

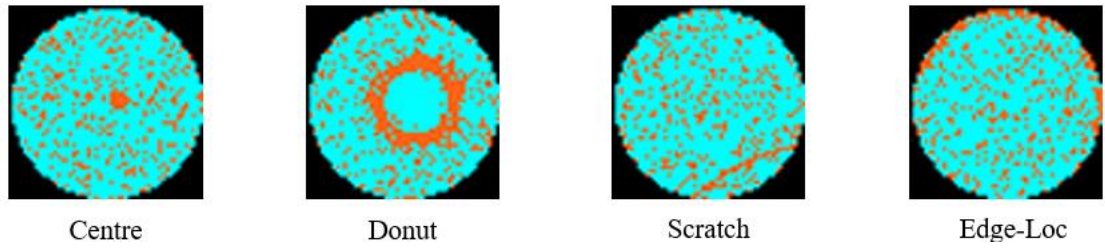


Figure 3.12: WBM images after colour encoding.

3.3.7 Image Processing - Resizing

In the pre-processing workflow for the YOLO model, it is crucial to ensure that each colour-encoded image is resized to dimensions that are a multiple of 32, due to the architectural requirements of the YOLO neural network. Originally, the dataset images are 52×52 pixels, which does not meet this requirement. Therefore, each image is resized to 64×64 pixels. This size not only adheres to YOLO's requirement but also standardizes the input size for the neural network models, ensuring uniform dimensions across all input images. Such consistency is vital for effective training and consistent processing. The choice of 64×64 pixels as the target size strikes a balance between maintaining high resolution for feature detection and managing the computational load efficiently. Larger images contain more pixels and hence more data for the model to process, which can slow down training and require more memory. The 64×64 dimension is manageable for most computational setups and provides enough detail for model training without overwhelming the system's capacity.

3.3.8 Directory Structure and Image Storage

Finally, the processed images are organized into directories corresponding to their labels and split categories (training, validation, test). This organization aids in the systematic management of data and simplifies the access and retrieval process during model training and testing phases. Each image is saved with a unique identifier that facilitates easy tracking and referencing. The data pre-processing steps outlined ensure

that the wafer map dataset is transformed from raw numerical grids into a structured, clean format ready for deep learning applications. These steps are critical for ensuring that the models developed are robust.

3.4 Model Training

In this study, YOLOv8 is employed for the classification of mixed-type WBM defect patterns. YOLOv8 is the recent version of YOLO algorithm, enhancing its capabilities with various modifications that improve both performance and processing speed. This version includes a range of models: nano (YOLOv8n), small (YOLOv8s), medium (YOLOv8m), large (YOLOv8l), and extra-large (YOLOv8x). In this study, all variants of YOLOv8 pretrained models are employed for comparison.

The YOLOv8 model is structured around three primary sections: the backbone, neck, and head, which are tasked with feature extraction, the fusion of multiple features, and outputting predictions, respectively. For extracting features at various scales, it incorporates the C2f and Spatial Pyramid Pooling Fast (SPPF) modules. The C2f module, an adaptation of the original C3 module, reduces complexity by eliminating a convolutional layer, thus making the model lighter while integrating the Efficient Layer Aggregation Network (ELAN) structure from YOLOv7 to improve gradient flow through bottleneck modules. Meanwhile, the SPPF module refines feature fusion by simplifying the traditional Spatial Pyramid Pooling (SPP) configuration (Zhai et al., 2023).

For multi-scale fusion, YOLOv8 employs a combination of Feature Pyramid Network (FPN) and Path Aggregation Network (PAN), which allows for effective bidirectional integration of both low and high-level features. This integration boosts the detection of objects across various sizes by enhancing features with smaller receptive fields (Zhai et al., 2023).

In the detection layer, YOLOv8 shifts from an anchor-based to an anchor-free approach, eliminating Intersection Over Union (IOU) matching and single-side scale assignment. It adopts a task-aligned assigner for more accurate sample matching. The model outputs predictions for various target sizes using multi-scale features, ensuring precise detection across different scales (Zhai et al., 2023).

Table 3.2: Hyperparameters for model training.

Hyperparameters	Description	Values
Epochs	Number of full passes through entire training dataset.	150
Batch size	Number of training examples utilized in one iteration.	16
Image size	Dimensions to which input images are resized.	64
Dropout rate	Regularization technique used to reduce overfitting in neural networks by setting a fraction of neurons' activations to zero.	0.15
Patience	Number of epochs with no improvement after which training will be stopped to reduce overfitting.	50
Optimizer	Optimization algorithm used to update network weights and biases during training to minimize loss function.	SGD

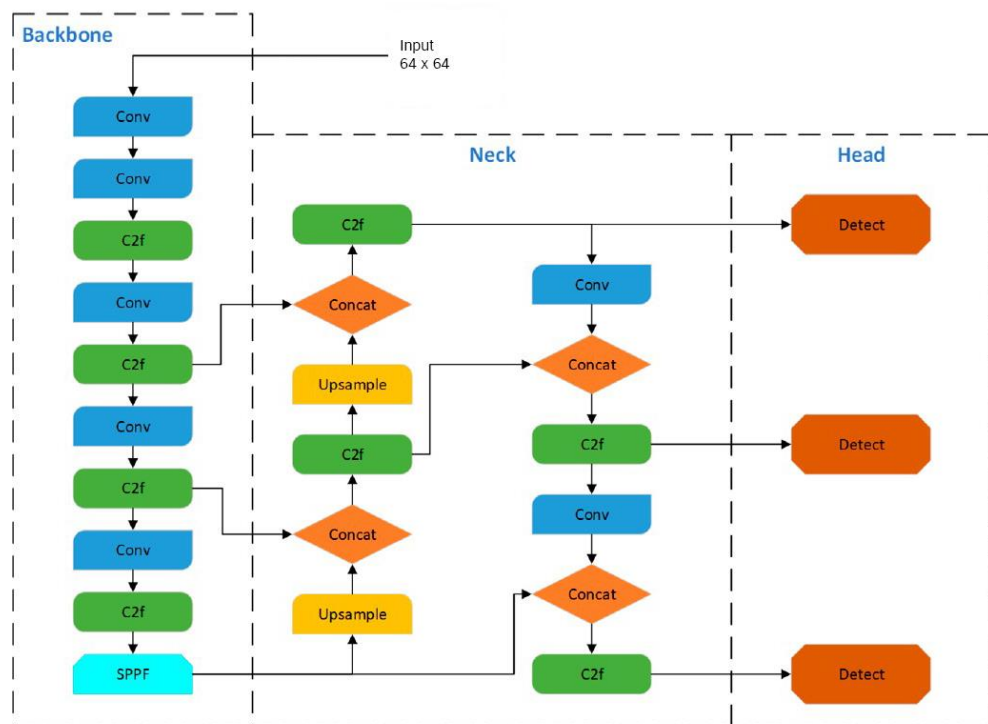


Figure 3.13: YOLOv8 network architecture diagram (Zhai et al., 2023).

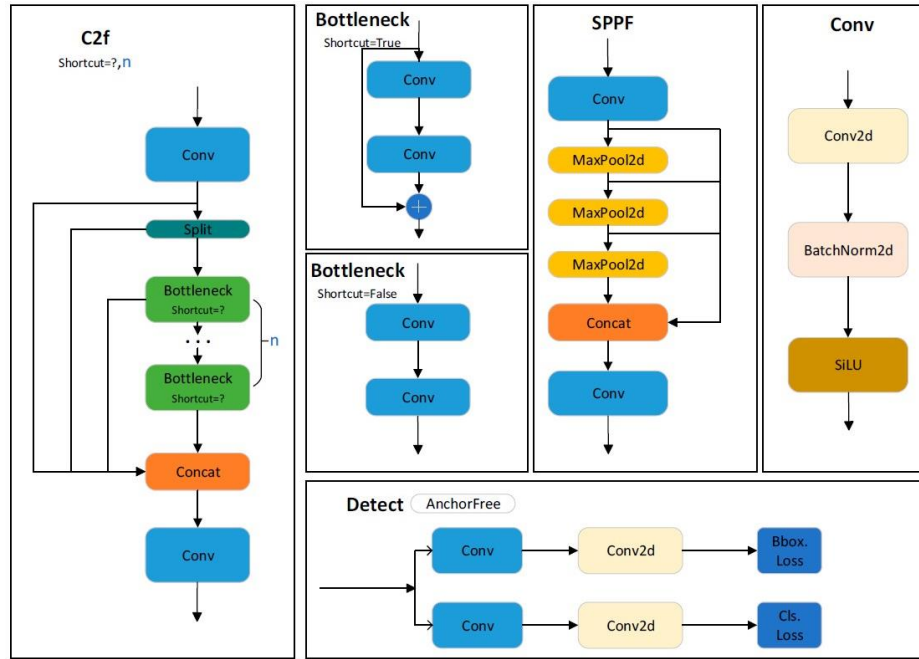


Figure 3.14: Detailed module in the YOLOv8 network (Zhai et al., 2023).

Table 3.3: Network parameters of YOLOv8n Pretrained Classify model (Ultralytics, 2024).

Layers	From	n	Params	Module	Arguments
0	-1	1	464	Conv	[3, 16, 3, 2]
1	-1	1	4672	Conv	[16, 32, 3, 2]
2	-1	1	7360	C2f	[32, 32, 1, True]
3	-1	1	18560	Conv	[32, 64, 3, 2]
4	-1	2	49664	C2f	[64, 64, 2, True]
5	-1	1	73984	Conv	[64, 128, 3, 2]
6	-1	2	197632	C2f	[128, 128, 2, True]
7	-1	1	295424	Conv	[128, 256, 3, 2]
8	-1	1	460288	C2f	[256, 256, 1, True]
9	-1	1	378918	Classify	[256, 38]
YOLOv8n-cls summary: 99 layers, 1486966 parameters, 1486966 gradients, 3.4 GFLOPs					

Table 3.4: Network parameters of YOLOv8s Pretrained Classify model
(Ultralytics, 2024).

Layers	From	n	Params	Module	Arguments
0	-1	1	928	Conv	[3, 32, 3, 2]
1	-1	1	18560	Conv	[32, 64, 3, 2]
2	-1	1	29056	C2f	[64, 64, 1, True]
3	-1	1	73984	Conv	[64, 128, 3, 2]
4	-1	2	197632	C2f	[128, 128, 2, True]
5	-1	1	295424	Conv	[128, 256, 3, 2]
6	-1	2	788480	C2f	[256, 256, 2, True]
7	-1	1	1180672	Conv	[256, 512, 3, 2]
8	-1	1	1838080	C2f	[512, 512, 1, True]
9	-1	1	706598	Classify	[512, 38]
YOLOv8s-cls summary: 99 layers, 5129414 parameters, 5129414 gradients, 12.6 GFLOPs					

Table 3.5: Network parameters of YOLOv8m Pretrained Classify model
(Ultralytics, 2024).

Layers	From	n	Params	Module	Arguments
0	-1	1	1392	Conv	[3, 48, 3, 2]
1	-1	1	41664	Conv	[48, 96, 3, 2]
2	-1	2	111360	C2f	[96, 96, 2, True]
3	-1	1	166272	Conv	[96, 192, 3, 2]
4	-1	4	813312	C2f	[192, 192, 4, True]
5	-1	1	664320	Conv	[192, 384, 3, 2]
6	-1	4	3248640	C2f	[384, 384, 4, True]
7	-1	1	2655744	Conv	[384, 768, 3, 2]
8	-1	2	7084032	C2f	[768, 768, 2, True]
9	-1	1	1034278	Classify	[768, 38]
YOLOv8m-cls summary: 141 layers, 15821014 parameters, 15821014 gradients, 41.9 GFLOPs					

Table 3.6: Network parameters of YOLOv8l Pretrained Classify model
(Ultralytics, 2024).

Layers	From	n	Params	Module	Arguments
0	-1	1	1856	Conv	[3, 64, 3, 2]
1	-1	1	73984	Conv	[64, 128, 3, 2]
2	-1	3	279808	C2f	[128, 128, 3, True]
3	-1	1	295424	Conv	[128, 256, 3, 2]
4	-1	6	2101248	C2f	[256, 256, 6, True]
5	-1	1	1180672	Conv	[256, 512, 3, 2]
6	-1	6	8396800	C2f	[512, 512, 6, True]
7	-1	1	4720640	Conv	[512, 1024, 3, 2]
8	-1	3	17836032	C2f	[1024, 1024, 3, True]
9	-1	1	1361958	Classify	[1024, 38]
YOLOv8l-cls summary: 183 layers, 36248422 parameters, 36248422 gradients, 99.2 GFLOPs					

Table 3.7: Network parameters of YOLOv8x Pretrained Classify model
(Ultralytics, 2024).

Layers	From	n	Params	Module	Arguments
0	-1	1	2320	Conv	[3, 80, 3, 2]
1	-1	1	115520	Conv	[80, 160, 3, 2]
2	-1	3	436800	C2f	[160, 160, 3, True]
3	-1	1	461440	Conv	[160, 320, 3, 2]
4	-1	6	3281920	C2f	[320, 320, 6, True]
5	-1	1	1844480	Conv	[320, 640, 3, 2]
6	-1	6	13117440	C2f	[640, 640, 6, True]
7	-1	1	7375360	Conv	[640, 1280, 3, 2]
8	-1	3	27865600	C2f	[1280, 1280, 3, True]
9	-1	1	1689638	Classify	[1280, 38]
YOLOv8x-cls summary: 183 layers, 56190518 parameters, 56190518 gradients, 154.3 GFLOPs					

3.5 Evaluation of Model Performance

The classification performance of the trained model is assessed using four key metrics: accuracy, precision, recall, and F1-score. These metrics are derived from four elements in the confusion matrix: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Table 3.8 summarizes the formulae used to calculate each metric.

Table 3.8: Calculation formulae for model performance metric.

Metric	Calculation Formula
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN} \times 100\%$
Precision	$\frac{TP}{TP + FP} \times 100\%$
Recall	$\frac{TP}{TP + FN} \times 100\%$
F1-score	$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \times 100\%$

CHAPTER 4: RESULTS

Table 4.1: Class-wise accuracy and precision of YOLOv8 models on test set.

Defect Class	Accuracy (%)					Precision (%)				
	YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x	YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x
Normal	99.68	99.68	100.00	99.68	99.68	99.68	99.68	100.00	99.68	99.68
Center	97.62	97.95	98.29	97.95	97.95	97.62	97.95	98.29	98.28	98.28
Donut	96.58	96.88	97.49	96.88	97.19	96.58	96.88	97.49	96.88	97.19
Edge_Loc	96.75	96.12	97.07	96.43	96.75	98.35	98.02	98.68	98.34	98.35
Edge_Ring	97.85	97.26	97.55	97.55	97.55	97.85	97.26	97.55	97.55	97.55
Loc	97.81	97.80	99.63	99.63	100.00	97.81	98.16	99.63	99.63	100.00
Near_Full	90.74	89.09	96.00	92.00	90.38	90.74	89.09	97.96	97.87	94.00
Scratch	96.49	96.49	96.49	96.83	97.17	96.49	96.49	96.49	96.83	97.17
Random	97.49	97.07	99.17	97.93	97.51	100.00	100.00	99.58	98.75	99.16
C_EL	95.08	95.10	96.04	96.04	96.36	97.64	97.32	98.31	98.31	98.64
C_ER	95.93	95.92	96.92	96.92	97.25	95.93	96.25	96.92	96.92	97.25
C_L	96.25	96.86	98.10	97.78	97.16	97.78	98.40	99.04	99.04	98.72
C_S	95.54	95.24	96.46	95.85	96.15	95.54	95.24	96.46	95.85	96.15
D_EL	96.62	96.61	98.29	97.27	97.94	98.28	98.62	99.65	99.30	100.00
D_L	95.95	96.87	97.17	97.17	97.17	98.40	99.04	99.36	99.36	99.36
ER_L	96.14	95.77	95.76	97.15	96.80	96.14	96.45	96.79	97.50	97.49
EL_S	96.66	96.95	96.95	97.55	97.55	99.07	99.38	99.38	100.00	100.00
ER_S	96.19	96.19	96.81	97.12	97.12	96.81	96.81	97.43	97.44	97.74
L_S	94.16	94.79	96.08	97.04	97.38	97.64	97.98	98.33	98.99	98.67

D_ER	97.17	98.41	99.04	98.72	98.41	97.17	98.41	99.04	98.72	98.41
D_S	96.08	96.72	96.72	97.03	96.72	96.71	97.04	97.04	97.67	97.04
EL_L	95.53	96.46	96.47	97.09	96.77	97.39	98.04	97.73	98.68	98.36
C_EL_L	94.10	95.63	95.02	96.56	96.25	98.70	99.35	99.03	99.36	99.35
C_EL_S	95.72	96.38	97.20	96.86	97.19	99.32	99.49	99.49	99.66	99.66
C_ER_L	94.12	94.43	93.53	94.75	95.07	95.36	95.68	94.44	95.70	96.01
C_ER_S	93.49	94.72	95.03	94.41	94.72	93.79	95.03	95.35	94.72	95.03
C_L_S	95.82	96.14	97.11	96.47	96.49	99.00	99.01	99.02	98.69	98.37
D_EL_L	98.38	99.35	99.03	99.35	99.35	100.00	100.00	100.00	100.00	100.00
D_EL_S	92.40	92.45	94.21	94.79	95.40	98.38	97.76	98.72	99.36	99.36
D_L_S	93.79	94.14	95.16	95.50	95.16	97.49	97.50	97.86	97.87	97.86
D_ER_L	97.85	99.07	99.08	99.38	99.38	98.76	99.38	99.08	99.38	99.38
D_ER_S	93.44	93.77	94.97	95.27	95.27	94.32	94.06	94.97	95.27	95.27
EL_L_S	93.06	94.79	94.79	96.88	96.18	100.00	100.00	100.00	100.00	100.00
ER_L_S	92.91	92.91	93.97	95.36	95.04	98.50	98.50	98.51	99.26	98.53
C_L_EL_S	93.24	94.93	94.26	95.27	94.59	100.00	100.00	100.00	100.00	100.00
C_L_ER_S	91.32	91.91	91.61	92.56	91.67	96.93	97.59	97.26	97.61	96.62
D_L_EL_S	93.29	93.29	94.25	95.53	94.89	100.00	100.00	100.00	100.00	100.00
D_L_ER_S	94.74	95.03	95.38	96.35	95.70	96.64	97.29	96.98	97.64	97.31
Average (1 defect)	96.78	96.48	97.97	97.21	97.13	97.24	97.06	98.41	98.20	97.93
Average (2 defects)	95.95	96.30	96.99	97.13	97.14	97.27	97.61	98.11	98.29	98.30
Average (3 defects)	94.59	95.31	95.76	96.30	96.29	97.80	97.98	98.04	98.27	98.24
Average (4 defects)	93.15	93.79	93.87	94.92	94.21	98.39	98.72	98.56	98.81	98.48
Average (All 38)	95.42	95.77	96.50	96.66	96.56	97.55	97.71	98.21	98.32	98.21

Table 4.2: Class-wise recall and F1-score of YOLOv8 models on test set.

Defect Class	Recall (%)					F1-score (%)				
	YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x	YOLOv8n	YOLOv8s	YOLOv8m	YOLOv8l	YOLOv8x
Normal	100.00	100.00	100.00	100.00	100.00	99.84	99.84	100.00	99.84	99.84
Center	100.00	100.00	100.00	99.65	99.65	98.80	98.97	99.14	98.96	98.96
Donut	100.00	100.00	100.00	100.00	100.00	98.26	98.42	98.73	98.42	98.57
Edge_Loc	98.35	98.02	98.35	98.02	98.35	98.35	98.02	98.51	98.18	98.35
Edge_Ring	100.00	100.00	100.00	100.00	100.00	98.91	98.61	98.76	98.76	98.76
Loc	100.00	99.63	100.00	100.00	100.00	98.89	98.89	99.81	99.81	100.00
Near_Full	100.00	100.00	97.96	93.88	95.92	95.15	94.23	97.96	95.83	94.95
Scratch	100.00	100.00	100.00	100.00	100.00	98.21	98.21	98.21	98.39	98.57
Random	97.49	97.07	99.58	99.16	98.33	98.73	98.51	99.58	98.96	98.74
C_EL	97.32	97.65	97.65	97.65	97.65	97.48	97.49	97.98	97.98	98.15
C_ER	100.00	99.65	100.00	100.00	100.00	97.92	97.92	98.43	98.43	98.61
C_L	98.40	98.40	99.04	98.72	98.40	98.09	98.40	99.04	98.88	98.56
C_S	100.00	100.00	100.00	100.00	100.00	97.72	97.56	98.20	97.88	98.04
D_EL	98.28	97.94	98.63	97.94	97.94	98.28	98.28	99.14	98.62	98.96
D_L	97.47	97.78	97.78	97.78	97.78	97.93	98.41	98.56	98.56	98.56
ER_L	100.00	99.27	98.91	99.64	99.27	98.03	97.84	97.83	98.56	98.37
EL_S	97.55	97.55	97.55	97.55	97.55	98.30	98.45	98.45	98.76	98.76
ER_S	99.34	99.34	99.34	99.67	99.34	98.06	98.06	98.38	98.54	98.54
L_S	96.35	96.68	97.67	98.01	98.67	96.99	97.32	98.00	98.50	98.67
D_ER	100.00	100.00	100.00	100.00	100.00	98.56	99.20	99.52	99.36	99.20

D_S	99.32	99.66	99.66	99.32	99.66	98.00	98.33	98.33	98.49	98.33
EL_L	98.03	98.36	98.69	98.36	98.36	97.71	98.20	98.21	98.52	98.36
C_EL_L	95.28	96.23	95.91	97.17	96.86	96.96	97.76	97.44	98.25	98.09
C_EL_S	96.36	96.85	97.68	97.19	97.52	97.82	98.15	98.58	98.41	98.58
C_ER_L	98.63	98.63	98.97	98.97	98.97	96.97	97.13	96.66	97.31	97.47
C_ER_S	99.65	99.65	99.65	99.65	99.65	96.63	97.29	97.45	97.12	97.29
C_L_S	96.75	97.08	98.05	97.73	98.05	97.87	98.03	98.53	98.21	98.21
D_EL_L	98.38	99.35	99.03	99.35	99.35	99.18	99.68	99.51	99.68	99.68
D_EL_S	93.83	94.44	95.37	95.37	95.99	96.05	96.08	97.02	97.32	97.65
D_L_S	96.11	96.47	97.17	97.53	97.17	96.80	96.98	97.52	97.70	97.52
D_ER_L	99.07	99.69	100.00	100.00	100.00	98.91	99.53	99.54	99.69	99.69
D_ER_S	99.01	99.67	100.00	100.00	100.00	96.61	96.78	97.42	97.58	97.58
EL_L_S	93.06	94.79	94.79	96.88	96.18	96.40	97.33	97.33	98.41	98.05
ER_L_S	94.24	94.24	95.32	96.04	96.40	96.32	96.32	96.89	97.62	97.45
C_L_EL_S	93.24	94.93	94.26	95.27	94.59	96.50	97.40	97.04	97.58	97.22
C_L_ER_S	94.04	94.04	94.04	94.70	94.70	95.46	95.78	95.62	96.13	95.65
D_L_EL_S	93.29	93.29	94.25	95.53	94.89	96.53	96.53	97.04	97.71	97.38
D_L_ER_S	97.96	97.62	98.30	98.64	98.30	97.30	97.45	97.64	98.14	97.80
Average (1 defect)	99.54	99.41	99.54	98.97	99.14	98.35	98.19	98.97	98.57	98.53
Average (2 defects)	98.62	98.64	98.84	98.82	98.82	97.93	98.11	98.47	98.54	98.55
Average (3 defects)	96.70	97.26	97.66	97.99	98.01	97.21	97.59	97.82	98.11	98.10
Average (4 defects)	94.63	94.97	95.21	96.03	95.62	96.45	96.79	96.84	97.39	97.01
Average (All 38)	97.81	98.00	98.25	98.30	98.30	97.65	97.83	98.21	98.29	98.24

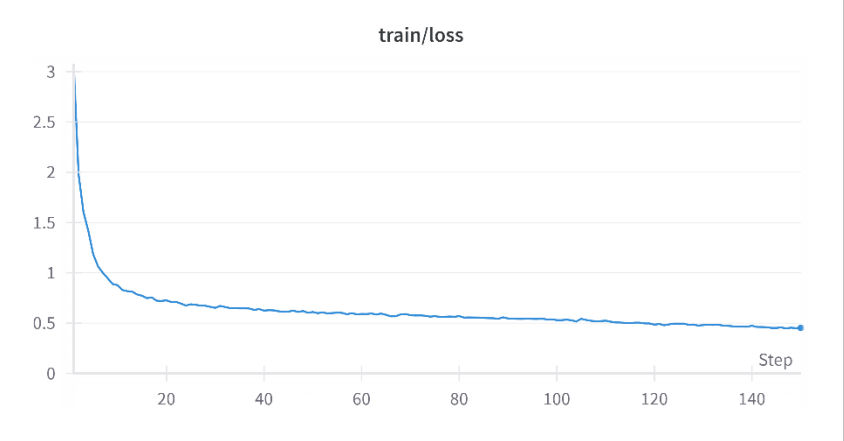
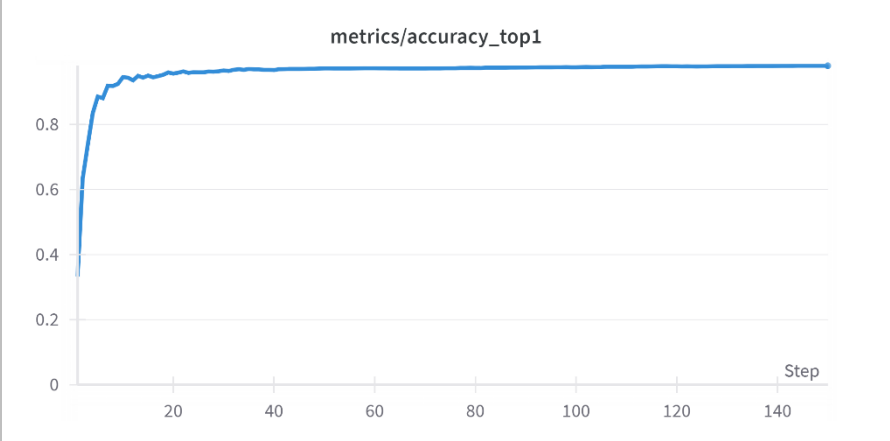
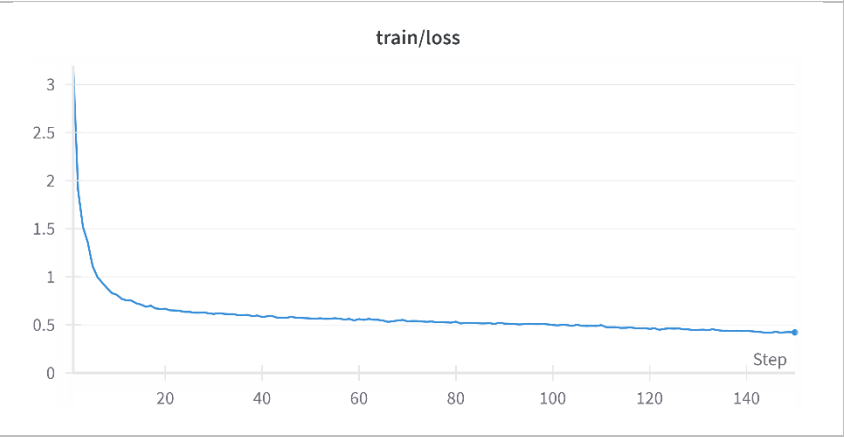
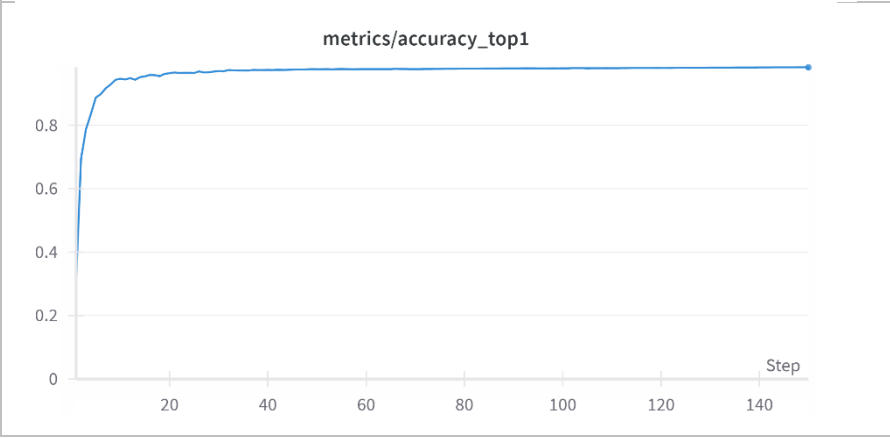
Table 4.3: Top-1 accuracy and inference time of YOLOv8 models on test set, and comparison with YOLO models in published literature.

Model	Top-1 Accuracy (%)	Inference Time (ms)
YOLOv8n	97.7	0.4
YOLOv8s	97.9	0.4
YOLOv8m	98.2	0.5
YOLOv8l	98.4	0.7
YOLOv8x	98.3	0.7
YOLOv3-tiny (Shinde et al., 2022)	82.8	35
YOLOv3 (Shinde et al., 2022)	94.4	20
YOLOv4 (Shinde et al., 2022)	95.7	18

Table 4.4: Comparison of YOLOv8 classification performance against existing models on MixedWM38 dataset.

Model	Accuracy (%)	Precision (%)	Recall (%)	Parameters	FLOPS	Epochs
YOLOv8n	97.7	97.55	97.81	2.71 M	4.4 G	150
YOLOv8s	97.9	97.71	98.00	5.13 M	12.6 G	150
YOLOv8m	98.2	98.21	98.25	15.82 M	41.9 G	150
YOLOv8l	98.4	98.32	98.30	36.24 M	99.2 G	150
YOLOv8x	98.3	98.21	98.30	56.2 M	154.3 G	150
WSCN (Nag et al., 2022)	98.2	98.08	98.18	0.09 M	0.2 M	150
WM-PeleeNet (Yu et al., 2022)	97.5	-	-	0.169 M	0.316 G	50
DCNet (Wang et al., 2020)	93.2	94.00	95.00	2.3 M	1.4 G	4000

Table 4.5: Learning curve and top-1 accuracy curve of YOLOv8 models on training set.

Model	Learning Curve on Training Set	Top-1 Accuracy Curve on Training Set
YOLOv8n		
YOLOv8s		

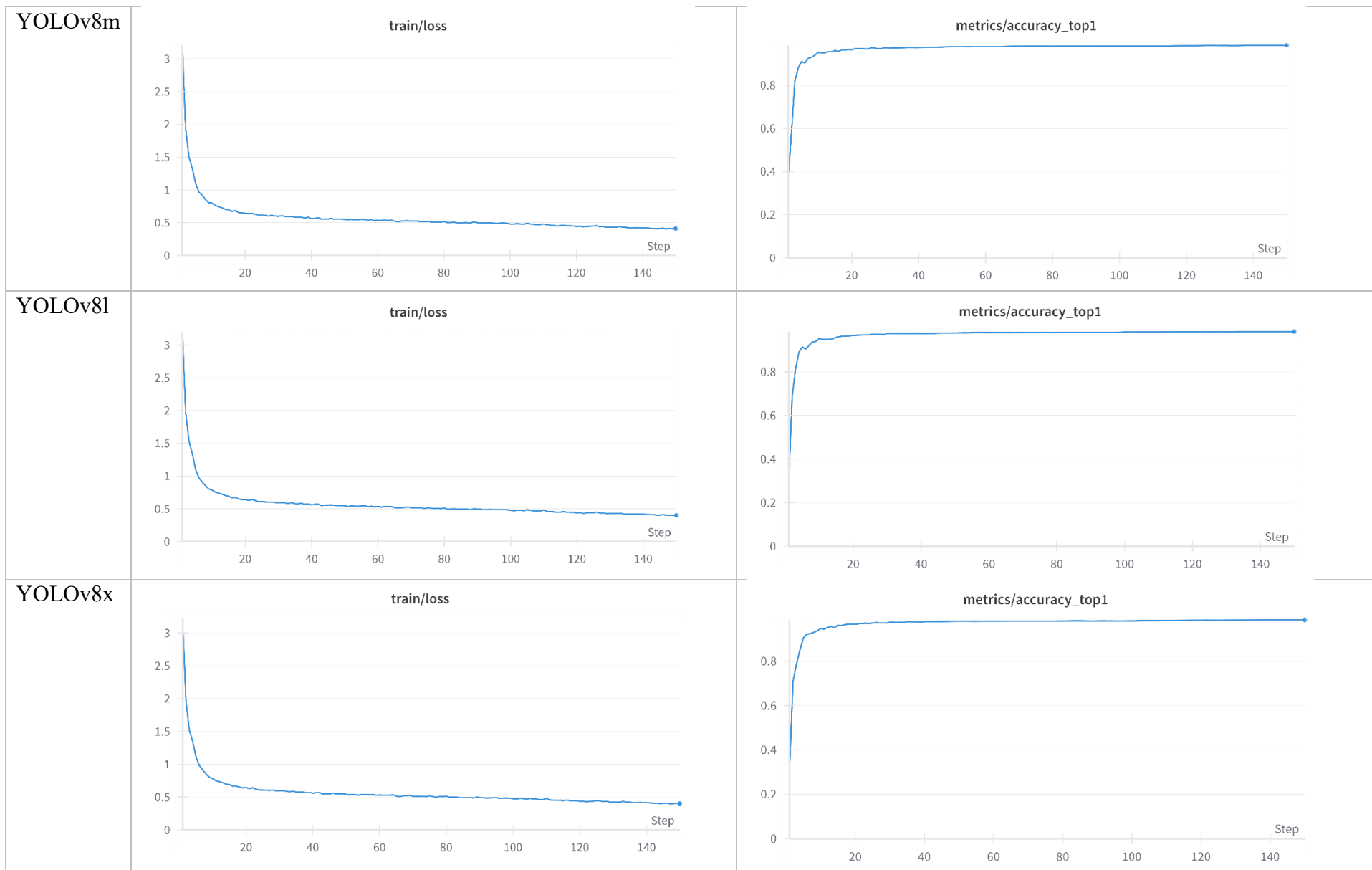
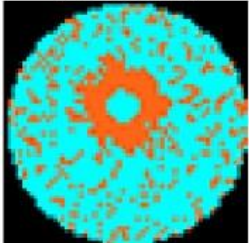
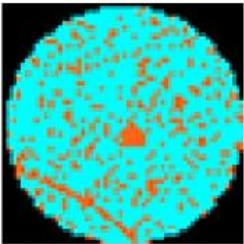
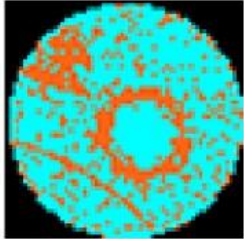
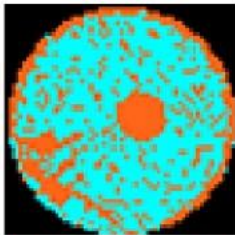


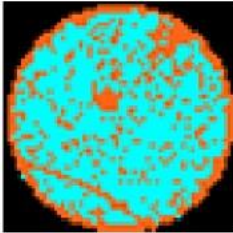
Table 4.6: Qualitative results on test set using YOLOv8l pretrained Classify model.

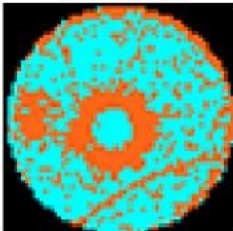
No.	Screenshot	Input	Output
1.	<pre> # Load the image image_path = "/kaggle/working/data/test/Donut/10460.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f'Predicted: {top_class} ({top_probability:.2f})', xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/Donut/10460.png: 64x64 Donut 0.99, D_EL 0.01, D_L 0.00, D_S 0.00, Center 0.00, 7.0ms Speed: 1.5ms preprocess, 7.0ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: Donut (0.99)</p>	<p>A WBM image containing “Donut” defect with index 10460 located within the test dataset.</p>	<p>Top Prediction: Donut Probability: 0.99</p> <p>Other predictions with lower probabilities: D_EL: 0.01 D_L: 0.00 D_S: 0.00 Center: 0.00</p>

2.	<pre> # Load the image image_path = "/kaggle/working/data/test/C_S/9185.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/C_S/9185.png: 64x64 C_S 0.98, C_EL_S 0.01, C_L_S 0.00, D_S 0.00, Scratch 0.00, 8.9ms Speed: 1.7ms preprocess, 8.9ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: C_S (0.98)</p>	<p>A WBM image containing “C_S” defect with index 9185 located within the test dataset.</p>	<p>Top Prediction: C_S Probability: 0.98</p> <p>Other predictions with lower probabilities: C_EL_S: 0.01 C_L_S: 0.00 D_S: 0.00 Scratch: 0.00</p>
----	---	---	--

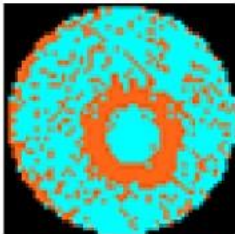
3.	<pre> # Load the image image_path = "/kaggle/working/data/test/D_L_S/3206.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/D_L_S/3206.png: 64x64 D_L_S 0.99, D_L_EL_S 0.01, D_L 0.00, C_L_S 0.00, L_S 0.00, 6.8ms Speed: 1.4ms preprocess, 6.8ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: D_L_S (0.99)</p>	<p>A WBM image containing “D_L_S” defect with index 3206 located within the test dataset.</p>	<p>Top Prediction: D_L_S Probability: 0.99</p> <p>Other predictions with lower probabilities: D_L_EL_S: 0.01 D_L: 0.00 C_L_S: 0.00 L_S: 0.00</p>
----	---	---	--

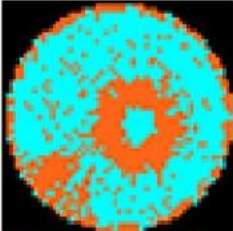
<p>4.</p>	<pre> # Load the image image_path = "/kaggle/working/data/test/C_L_EL_S/9626.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/C_L_EL_S/9626.png: 64x64 C_L_EL_S 1.00, D_L_EL_S 0.00, C_EL_L 0.00, C_L_ER_S 0.00, C_L_S 0.00, 7.7ms Speed: 1.6ms preprocess, 7.7ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: C_L_EL_S (1.00)</p>	<p>A WBM image containing “C_L_EL_S” defect with index 9626 located within the test dataset.</p>	<p>Top Prediction: C_L_EL_S Probability: 1.00</p> <p>Other predictions with lower probabilities:</p> <p>D_L_EL_S: 0.00 C_EL_L: 0.00 C_L_ER_S: 0.00 C_L_S: 0.00</p>
-----------	---	--	--

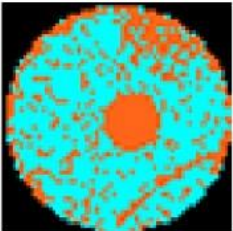
<p>5.</p>	<pre> # Load the image image_path = "/kaggle/working/data/test/C_L_ER_S/10803.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/C_L_ER_S/10803.png: 64x64 C_L_ER_S 0.98, C_L_EL_S 0.02, C_ER_L 0.00, D_L_ER_S 0.00, ER_L_S 0.00, 8.5ms Speed: 1.6ms preprocess, 8.5ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: C_L_ER_S (0.98)</p>	<p>A WBM image containing “C_L_ER_S” defect with index 10803 located within the test dataset.</p>	<p>Top Prediction: C_L_ER_S Probability: 0.98</p> <p>Other predictions with lower probabilities: C_L_EL_S: 0.02 C_ER_L: 0.00 D_L_ER_S: 0.00 ER_L_S: 0.00</p>
-----------	--	---	--

<p>6.</p>	<pre> # Load the image image_path = "/kaggle/working/data/test/D_L_EL_S/10126.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/D_L_EL_S/10126.png: 64x64 D_L_EL_S 1.00, EL_L_S 0.00, D_L_S 0.00, D_EL_L 0.00, C_L_EL_S 0.00, 6.9ms Speed: 1.5ms preprocess, 6.9ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: D_L_EL_S (1.00)</p>	<p>A WBM image containing “D_L_EL_S” defect with index 10126 located within the test dataset.</p>	<p>Top Prediction: D_L_EL_S Probability: 1.00</p> <p>Other predictions with lower probabilities: EL_L_S: 0.00 D_L_S: 0.00 D_EL_L: 0.00 C_L_EL_S: 0.00</p>
-----------	---	---	---

<p>7.</p>	<pre> # Load the image image_path = "/kaggle/working/data/test/D_L_ER_S/1880.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/D_L_ER_S/1880.png: 64x64 D_L_ER_S 0.99, D_L_EL_S 0.01, D_ER_L 0.00, C_L_ER_S 0.00, D_L_S 0.00, 6.9ms Speed: 1.5ms preprocess, 6.9ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: D_L_ER_S (0.99)</p>	<p>A WBM image containing “D_L_ER_S” defect with index 1880 located within the test dataset.</p>	<p>Top Prediction: D_L_ER_S Probability: 0.99</p> <p>Other predictions with lower probabilities:</p> <p>D_L_EL_S: 0.01 D_ER_L: 0.00 C_L_ER_S: 0.00 D_L_S: 0.00</p>
-----------	---	--	--

8.	<pre> # Load the image image_path = "/kaggle/working/data/test/D_EL/349.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/D_EL/349.png: 64x64 D_EL 1.00, D_EL_S 0.00, D_EL_L 0.00, Edge_Loc 0.00, C_EL 0.00, 6.9ms Speed: 1.4ms preprocess, 6.9ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: D_EL (1.00)</p>	<p>A WBM image containing “D_EL” defect with index 349 located within the test dataset.</p>	<p>Top Prediction: D_EL Probability: 1.00</p> <p>Other predictions with lower probabilities:</p> <p>D_EL_S: 0.00 D_EL_L: 0.00 Edge_Loc: 0.00 C_EL: 0.00</p>
----	--	---	---

<p>9.</p>	<pre> # Load the image image_path = "/kaggle/working/data/test/D_ER_L/11340.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f"Predicted: {top_class} ({top_probability:.2f})", xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/D_ER_L/11340.png: 64x64 D_ER_L 1.00, D_EL_L 0.00, D_L_ER_S 0.00, C_ER_L 0.00, ER_L 0.00, 6.7ms Speed: 1.4ms preprocess, 6.7ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: D_ER_L (1.00)</p>	<p>A WBM image containing “D_ER_L” defect with index 11340 located within the test dataset.</p>	<p>Top Prediction: D_ER_L Probability: 1.00</p> <p>Other predictions with lower probabilities:</p> <p>D_EL_L: 0.00 D_L_ER_S: 0.00 C_ER_L: 0.00 ER_L: 0.00</p>
-----------	--	---	---

10.	<pre> # Load the image image_path = "/kaggle/working/data/test/C_L_EL_S/6855.png" image = cv2.imread(image_path) image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Predict with the model weights = "/kaggle/working/wafer_defects/EXP00014/weights/best.pt" model = YOLO(weights) results = model(image_path) # predict on the image # Access the probability tensor and convert it to a numpy array probs_tensor = results[0].probs.data probs = probs_tensor.cpu().numpy() # Access the class names class_names = results[0].names # Get the top prediction top_index = probs.argmax() top_class = class_names[top_index] top_probability = probs[top_index] # Plot the image plt.figure(figsize=(3, 3)) plt.imshow(image) plt.axis('off') # Annotate with top prediction below the figure plt.annotate(f'Predicted: {top_class} ({top_probability:.2f})', xy=(0.5, -0.1), xycoords='axes fraction', fontsize=12, ha='center', color='red') plt.title('YOLO Prediction') plt.show() </pre> <p>image 1/1 /kaggle/working/data/test/C_L_EL_S/6855.png: 64x64 C_L_EL_S 1.00, C_EL_S 0.00, C_L_S 0.00, O_L_EL_S 0.00, C_EL_L 0.00, 7.0ms Speed: 1.4ms preprocess, 7.0ms inference, 0.1ms postprocess per image at shape (1, 3, 64, 64)</p> <p>YOLO Prediction</p>  <p>Predicted: C_L_EL_S (1.00)</p>	<p>A WBM image containing “C_L_EL_S” defect with index 6855 located within the test dataset.</p>	<p>Top Prediction: C_L_EL_S Probability: 1.00</p> <p>Other predictions with lower probabilities:</p> <p>C_EL_S: 0.00 C_L_S: 0.00 D_L_EL_S: 0.00 C_EL_L: 0.00</p>
-----	---	--	--

CHAPTER 5: DISCUSSION

5.1 Framework and Configuration Overview

The experiments are conducted using the Ultralytics YOLOv8.2.18 framework, leveraging Python 3.10.13, PyTorch 2.1.2 deep learning library, and CUDA 11.2 on a Tesla P100-PCIE-16GB GPU available on Kaggle platform. The initial learning rate is set to 0.01. A starting rate of 0.01 is standard and provides a good balance between training speed and model convergence. The decay rate is set to 0.1, which means the learning rate will be reduced by a factor of 0.1 when there is no improvement in model convergence for 50 continuous epochs. This helps to fine-tune the model as it approaches convergence and avoid overfitting. The batch size is configured at 16. The size of 16 is a moderate choice that balances the memory usage and provides a good gradient estimate per iteration. The Stochastic Gradient Descent (SGD) optimizer is used for training the model. SGD is a robust optimizer that helps to minimize the loss function, aiming to find the best weights for the model. The training utilizes a standard categorical cross-entropy loss function. It works well with the SoftMax activation function in the output layer, which normalizes the class probabilities. This is crucial for multi-class classification tasks as it allows for a clear interpretation of the model's predictions as probabilities. Model performance is periodically evaluated on a validation set comprising 5,322 images, segmented into the same 38 classes as the training set. Training is conducted over 150 epochs with extensive logging and monitoring facilitated by Weights & Biases, a powerful tool for tracking experiments in machine learning. The best-performing model during training that has the highest validation performance is saved for further evaluation on the test set.

5.2 Comparative Analysis among YOLOv8 Variants

Table 4.1 and Table 4.2 present the class-wise accuracy, precision, recall, and F1-score results for each YOLOv8 variant. These models range from YOLOv8n (nano) to

YOLOv8x (extra-large) with increasing model complexities and computational demands. The evaluation of various YOLOv8 models helps to understand their effectiveness in classifying wafer defects. Generally, YOLOv8 models show remarkable classification accuracy, precision, recall and F1-score which indicate robust classification capabilities for wafer map defect patterns.

The YOLOv8 series expands in sophistication from the nano version, designed for environments with stringent computational constraints, to the extra-large version, which is intended for scenarios where prediction accuracy is paramount. From a technical standpoint, each variant of the YOLOv8 models, ranging from nano to extra-large, integrates an increasing number of layers and parameters, intended to enhance the model's ability to learn and generalize from complex spatial patterns in the WBM images. The deep learning approaches used by YOLOv8 rely on extracting both detailed texture features from pre-level convolution networks and higher-level information from the post-level convolutional layer. This dual approach helps in effectively capturing the spatial features of wafer maps, which are crucial for recognizing objects with geometrical deformations (Ren et al., 2017; Shinde et al., 2022).

Across these YOLOv8 variants, a general trend is observed where classification accuracy increases from YOLOv8n to YOLOv8l. However, there is a noticeable drop in classification accuracy when moving to YOLOv8x. This could potentially be attributed to overfitting, where the model's complexity captures noise along with the underlying data patterns, thus degrading its generalization capabilities on unseen data. The comparative decrease in performance from YOLOv8l to YOLOv8x suggests a critical point beyond which additional model complexity does not translate into better performance and may indeed hamper the model due to overfitting. Future work could explore regularization techniques that constrain model complexity to optimize the trade-off between accuracy and generalizability.

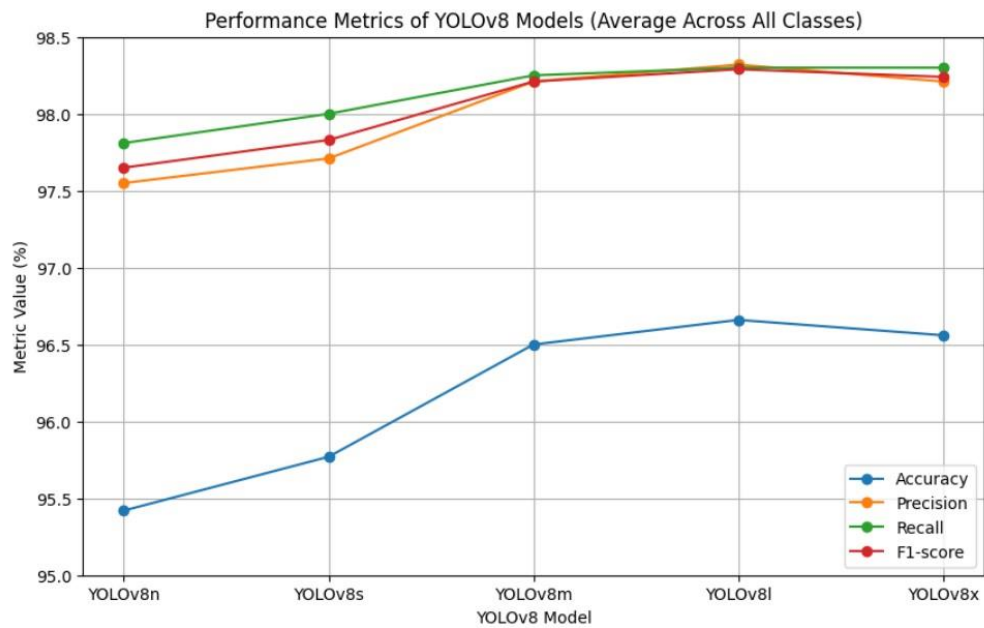


Figure 5.1: Average performance metrics of YOLOv8 models across all classes.

Table 5.1: Top-1 accuracy of YOLOv8 models on training and test sets.

Model	Top-1 Accuracy on Training Set (%)	Top-1 Accuracy on Test Set (%)
YOLOv8n	98.0	97.7
YOLOv8s	98.3	97.9
YOLOv8m	98.5	98.2
YOLOv8l	98.5	98.4
YOLOv8x	98.6	98.3

All versions of YOLOv8 demonstrate high accuracy on both training and test sets, indicating strong generalization capabilities on unseen data, as both training and test set accuracies are very close. There is a slight tendency for the models to perform marginally better on the training data compared to unseen test data, typically by 0.1% to 0.4%. This minor difference suggests a minimal presence of overfitting, where the models may slightly memorize training data specifics rather than fully generalize. YOLOv8l and YOLOv8x show the highest test set accuracies at 98.4% and 98.3% respectively, indicating effective generalization with larger model sizes. However, the comparative drop in accuracy from YOLOv8l to YOLOv8x highlights a critical consideration on the

potential risks of overfitting as model complexity increases. YOLOv8x, being the largest variant, may be more susceptible to capturing noise or irrelevant details from the training data, thus slightly diminishing its performance on unseen data. To further enhance robustness, exploring regularization techniques and increasing dataset diversity through augmentation could mitigate these slight overfitting tendencies. These methods aim to reduce the model's reliance on specific features of the training data and enhance its ability to generalize.

5.3 Comparative Analysis with Older YOLO Versions

The classification results for YOLOv8 variants are compared with those of older YOLO versions. In the literature published by Shinde et al. (2022), research was conducted on the WM-811K dataset containing single-type WBM defect patterns by using YOLOv3-tiny, YOLOv3, and YOLOv4 for classification. The published findings showed that YOLOv4 demonstrated superior classification accuracy than its predecessors. Consistent with the published findings, the YOLOv8 models in this study further outperform the older YOLO versions with YOLOv8l leading at 98.4% in terms of classification accuracy, emphasizing improvements in model architecture over time.

Additionally, the enhancement in model architecture heightens the effectiveness of the YOLOv8 variants in classifying defects on WBM images as indicated by the reduction in inference time. Inference time refers to the time taken to process a single image. As shown in Table 4.3, YOLOv8 models demonstrate remarkably lower inference times (0.4 to 0.7 milliseconds) compared to older YOLO models (18 to 35 milliseconds). Such improvements in processing speed, combined with increased accuracy, enables the YOLOv8 models to be well-suited for real-time applications where rapid and reliable defect detection is crucial. These attributes highlight the continual advancements in YOLO technology, emphasizing its capability to meet industrial demands for precision and speed.

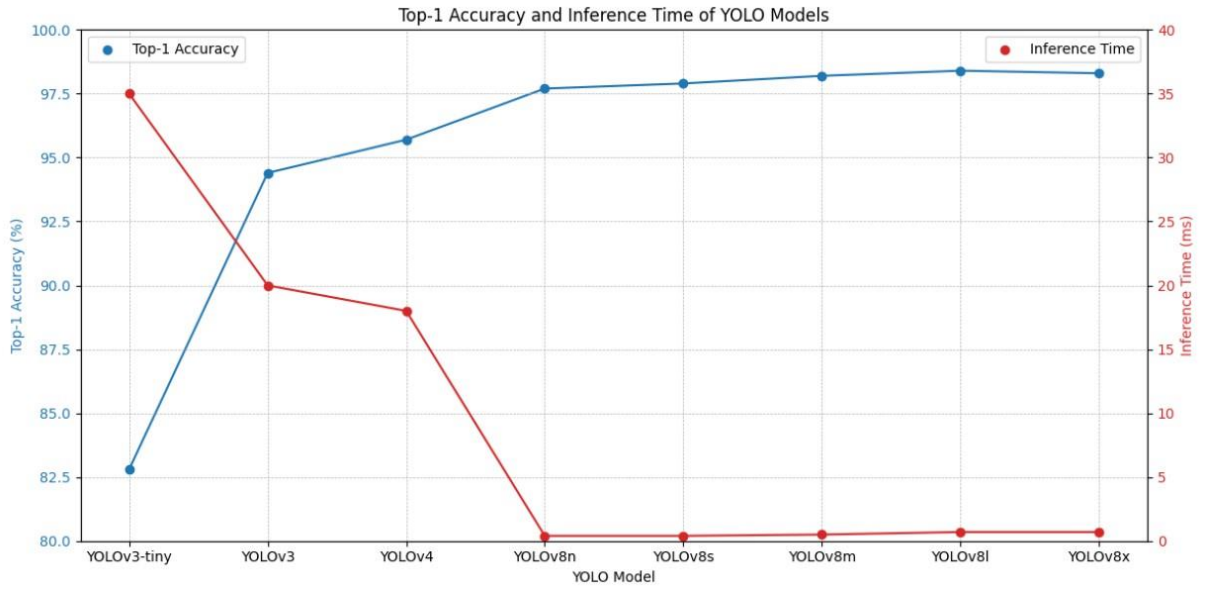


Figure 5.2: Top-1 accuracy and inference time of YOLO models.

5.4 Comparative Analysis with State-of-the-Art Models

The classification results for YOLOv8 variants are compared with those of previously published models such as WSCN (Nag et al., 2022), WM-PeleeNet (Yu et al., 2022), and DCNet (Wang et al., 2020). These comparisons are essential because WSCN, WM-PeleeNet and DCNet are trained on the same dataset MixedWM38 for the same objective, i.e., classification of mixed-type WBM defect patterns.

Table 4.4 provides a summary of overall classification accuracies known as top-1 accuracies, and further delves into the technical details such as parameter counts, floating-point operations per second (FLOPS), and number of training cycles (epoch). Although YOLOv8l has larger size and higher computational demand with 36.24 million parameters and 99.2 GFLOPS, the model achieves the highest top-1 accuracy of 98.4%, surpassing all the existing models including WSCN, WM-PeleeNet and DCNet. Moreover, the model's precision and recall metrics are consistently superior to those of existing models such as WSCN and DCNet, highlighting the robustness of YOLOv8l variant in diverse testing scenarios. These results emphasize the capability of YOLOv8

architectures to efficiently process and classify complex image data, enabling YOLOv8 to be highly effective tools in visual inspection tasks.

Furthermore, the average classification accuracy for single-type, two mixed-type, three mixed-type, and four mixed-type defect patterns using YOLOv8l variant are 97.21%, 97.13%, 96.3% and 94.92% respectively, as illustrated in Table 4.1 and Table 4.2. The model performance tends to decrease slightly as the model addresses more complex defect combinations (e.g., three mixed-type, four mixed-type), reflecting the increased challenge in classifying more complex patterns. However, YOLOv8l demonstrates minimal degradation with increasing defect complexity—a notable improvement over existing models such as WSCN (Nag et al., 2022).

All YOLOv8 models have been uniformly trained for 150 epochs, ensuring a consistent framework for comparing their performance metrics such as accuracy, precision, and recall. This uniformity in epochs across models allows for a direct comparison of other factors such as model complexity indicated by parameter counts and FLOPS on the final performance. In contrast, DCNet has been trained for an extensive 4000 epochs, which highlights its requirement for more iterations due to more complex architectures that need more time to converge. Training for too many epochs might lead to overfitting, especially if the model starts to learn not only the underlying pattern but also the noise within the training data, thereby reducing its ability to generalize. This might be an issue with DCNet, where despite the extensive training, the model performs lower than YOLOv8 variants. On the other hand, WM-PeleeNet's training for only 50 epochs indicates a faster convergence, possibly due to a simpler network architecture which involves only 0.169 million parameters.

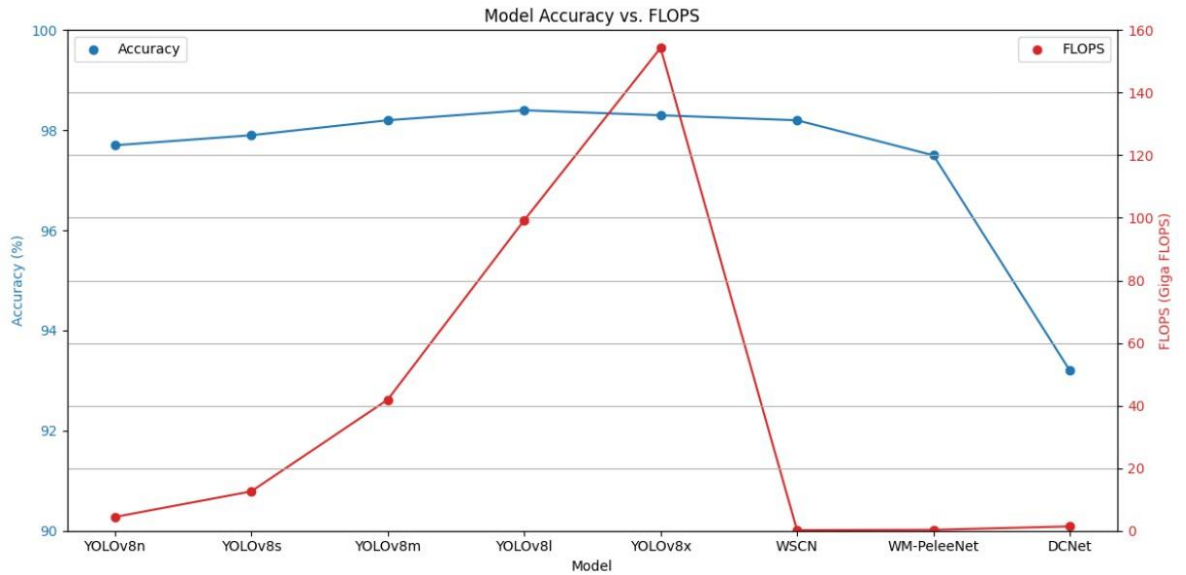


Figure 5.3: Model accuracy vs. FLOPS for YOLOv8 and state-of-the-art models.

5.5 Discrepancies in Model Performance Metrics

During the evaluation, it is noticed that there is a consistent difference between top-1 accuracy (overall accuracy) and the average accuracy across all classes. For instance, the top-1 accuracy for YOLOv8n model is 97.7% whereas the average accuracy across all classes for the same model is 95.42%. Understanding the discrepancy between these metrics is crucial for a comprehensive assessment of the model's performance, hence it warrants further discussion.

Top-1 accuracy measures how often the model correctly predicts the most likely class label for each input. This metric assesses the model's overall ability in identifying the most probable class from all available classes across all test images. A high top-1 accuracy indicates that the model performs well overall, but it does not reveal performance variations across different classes. On the other hand, average accuracy across all classes is calculated as the average of accuracies obtained for each class individually. This metric is crucial for understanding how the model performs across a diverse set of classes, particularly in scenarios where class distribution might not be uniform. It highlights how

some classes may achieve higher accuracy than others, potentially revealing underlying issues such as class imbalance or varying difficulty levels among the classes.

There are two potential reasons for discrepancy. First potential reason is class imbalance. Many real-world datasets exhibit some degree of class imbalance. This can skew the top-1 accuracy if a few classes dominate the dataset, as the model's performance on these classes disproportionately impacts the overall metric. Conversely, average accuracy across all classes is more sensitive to this imbalance as it equally weighs the accuracy of each class, revealing the true performance spread across all classes. Second potential reason is varying class difficulty. Certain classes may inherently be more challenging to predict due to subtle features, or because of their similarity to other classes. The model might achieve lower accuracy on these difficult classes, which would significantly impact the average accuracy across all classes but might not as severely affect the top-1 accuracy if these classes are less prevalent within the dataset (Ghosh et al., 2022).

To address these discrepancies, strategies such as re-weighting classes, and augmenting data for underrepresented classes can be employed to improve classification performance of more challenging classes. Such targeted improvements are essential for enhancing the model's robustness and fairness across various classes of input data. The next section examines the dataset distribution and identifies how underrepresented classes contributes to the discrepancies between top-1 accuracy and the average accuracy across all classes.

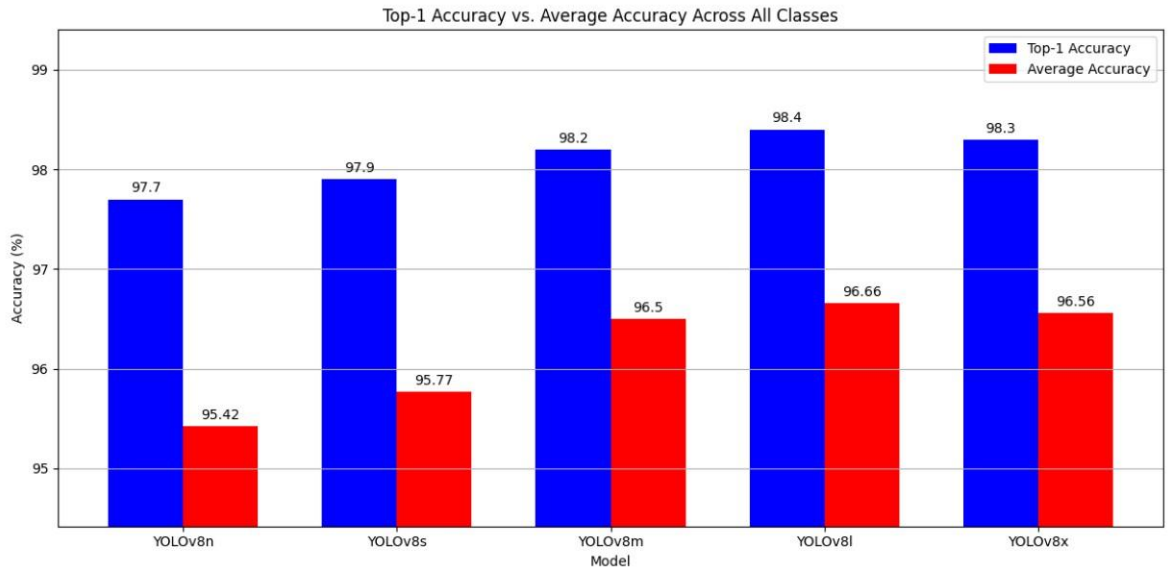


Figure 5.4: Top-1 accuracy vs. average accuracy across all classes for YOLOv8 models.

5.6 Performance Analysis for Specific Classes

The dataset distribution and performance metrics provided in Table 3.1, Table 4.1 and Table 4.2 reveal important insights into how the number of training examples per class can influence the YOLOv8 models' ability to learn and generalize across various defect classes. Most defect classes and their combinations in the dataset are represented by 1000 images. However, exceptions such as the Random (R) class with only 866 images and the Near-Full (NF) class with a significantly lower count of 149 images present challenges in model training due to underrepresentation. Underrepresented classes in the dataset might not be learned as effectively by the models during training. Consequently, the model has less ability to accurately generalize on unseen data, leading to poorer performance as indicated in the results. Additionally, it contributes to the discrepancies between top-1 accuracy and the average accuracy across all classes.

Random class shows relatively lower accuracy compared to most other classes, particularly in the YOLOv8n and YOLOv8s models. The reduced number of training images (866) for this class could be influencing this outcome by not providing the models

with sufficient data to effectively learn the variations within this defect class. High recall coupled with lower precision for the Random class suggests that while the models are able to detect the presence of these defects, they also tend to falsely identify other defects as Random. This could lead to a higher number of false positives, adversely affecting the precision.

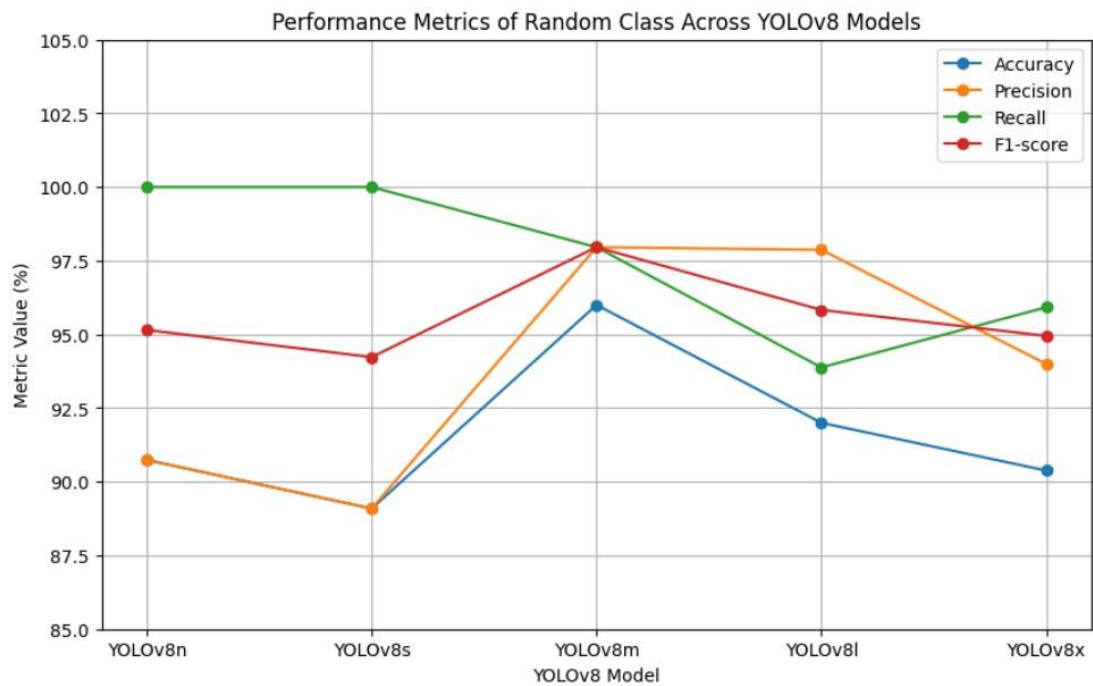


Figure 5.5: Performance metrics of Random class across YOLOv8 models.

Similarly, the Near-Full class demonstrates moderate accuracy but with notable inconsistencies across different YOLOv8 models. With only 149 images, this class is significantly underrepresented, which likely contributes to these fluctuations in performance, as the model has limited examples from which to learn. Near-Full class shows high precision but varying recall. This suggests that while the models can detect this defect class with a high level of accuracy when it is present, they are not consistently capturing all instances of the Near-Full class in the dataset.

These observations underscore the critical impact of balanced class representation in training datasets for deep learning models. The discrepancies in dataset distribution can

skew the model's learning, leading to biased predictions and variability in performance across different defect classes. This situation calls for strategies such as augmenting data for underrepresented classes and employing resampling techniques to balance the class distribution. Such measures would help in enhancing the model's robustness and ensuring fair and accurate performance across all classes.

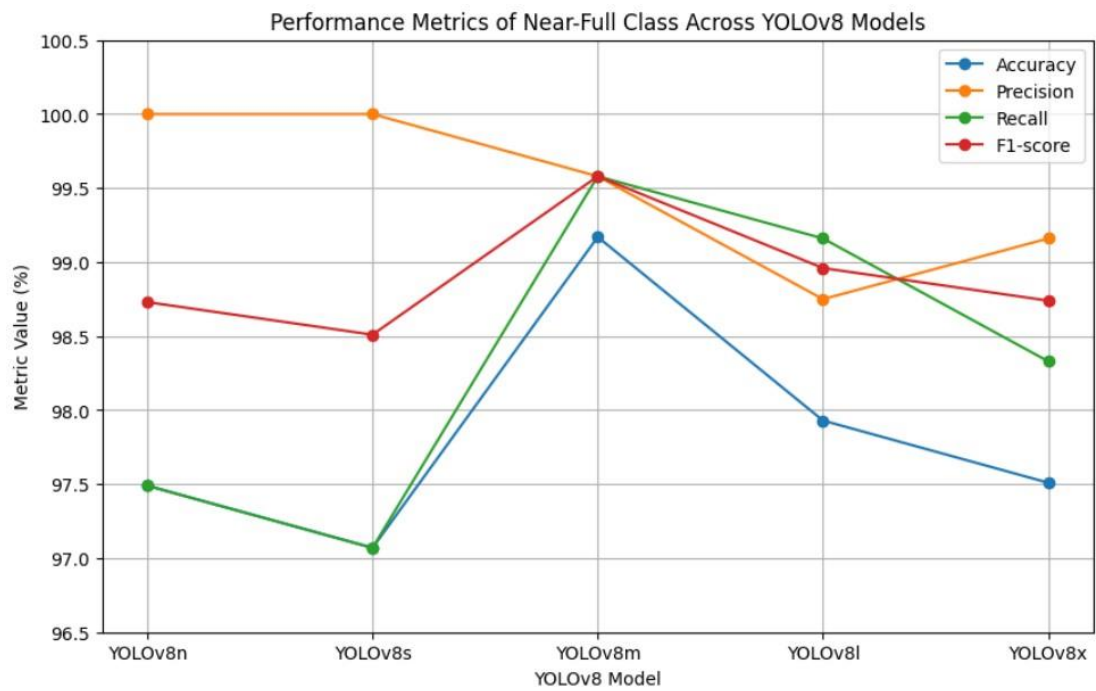


Figure 5.6: Performance metrics of Near-Full class across YOLOv8 models.

CHAPTER 6: CONCLUSION

WBM defect patterns are crucial for root cause analysis in semiconductor manufacturing, directly influencing product yield. Traditionally, the industry relied on manual inspections by domain experts, an inefficient method prone to human error. As a result, there has been a significant shift towards leveraging computer-aided methods, specifically machine learning, to automate wafer defect monitoring. While conventional machine learning approaches have been utilized, they often fall short in image processing and pattern recognition due to their reliance on manual feature extraction, leading to potential information distortion.

Deep learning emerges as a promising alternative, offering robust pattern recognition capabilities without the need for manual feature extraction. Among deep learning architectures, CNNs are noted for their effectiveness in classifying wafer defects, due to their robust algorithms and high accuracy. However, much of the existing literature has focused predominantly on single-type defect classification, which does not fully address the complexities encountered in real-world manufacturing scenarios where wafers often exhibit multiple types of defects simultaneously.

Recognizing this gap, this research utilized the recent version of YOLOv8 architecture to classify both single-type and mixed-type defects within the MixedWM38 dataset—a complex dataset consisting of 38 classes of WBM defect patterns. Among the various YOLOv8 variants examined, the YOLOv8l variant provided the best overall results. YOLOv8l achieved a classification accuracy of 98.4%, which surpassed all the existing models, indicating its superior ability to correctly classify WBM defect patterns. Despite its larger size and higher computational demand, YOLOv8l maintained a processing speed with an inference time of 0.7 milliseconds, making it suitable for real-time applications. Furthermore, while YOLOv8x showed signs of overfitting, YOLOv8l managed to strike a balance between model complexity and generalization, avoiding the

pitfalls of overfitting seen in more complex models. These advantages make YOLOv8l an ideal choice for automating wafer defect classification, providing both high accuracy and efficient processing capabilities.

To further improve the robustness of defect classification models, future research should focus on augmenting data for underrepresented classes and employing resampling techniques to balance class distribution. These strategies will enhance model performance across all defect classes, thereby supporting the semiconductor industry in achieving higher efficiency and improved product quality.

REFERENCES

- Batool, U., Shapiai, M. I., Tahir, M., Ismail, Z. H., Zakaria, N. J., & Elfakharany, A. (2021). A systematic review of deep learning for silicon wafer defect recognition. *IEEE Access*, 9, 116572–116593. <https://doi.org/10.1109/access.2021.3106171>
- Bhardwaj, R., Bandi, S., Purvesh, A., Aldar, P., & Borse, R. (2023). Semiconductor wafer defect detection using Deep Learning. *PriMera Scientific Engineering*. <https://doi.org/10.56831/psen-04-097>
- Byun, Y., & Baek, J.-G. (2020). Mixed pattern recognition methodology on wafer maps with pre-trained convolutional neural networks. *Proceedings of the 12th International Conference on Agents and Artificial Intelligence*. <https://doi.org/10.5220/0009177909740979>
- Cao, X., Zhang, F., Yi, C., Tang, K., Bian, T., & Yang, M. (2020). Wafer surface defect detection based on improved YOLOv3 Network. *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*. <https://doi.org/10.1109/icmcce51767.2020.00323>
- Chen, S.-H., Kang, C.-H., & Perng, D.-B. (2020). Detecting and measuring defects in wafer die using GAN and YOLOv3. *Applied Sciences*, 10(23), 8725. <https://doi.org/10.3390/app10238725>
- Cheon, S., Lee, H., Kim, C. O., & Lee, S. H. (2019). Convolutional neural network for wafer surface defect classification and the detection of unknown defect class. *IEEE Transactions on Semiconductor Manufacturing*, 32(2), 163–170. <https://doi.org/10.1109/tsm.2019.2902657>
- Choi, S., Xie, Q., Greenelch, N. G., Lee, H. J., Govindaraj, M., Jayaram, S., Pereira, M., Biswas, S., Bhamidipati, S., & Torunoglu, I. (2024). Fast and accurate automatic wafer defect detection and classification using machine learning based SEM image analysis. *Metrology, Inspection, and Process Control XXXVIII*. <https://doi.org/10.1117/12.3012184>
- Dehaerne, E., Dey, B., Halder, S., & De Gendt, S. (2023). Optimizing YOLOv7 for semiconductor defect detection. *Metrology, Inspection, and Process Control XXXVII*. <https://doi.org/10.1117/12.2657564>
- Devika, B., & George, N. (2019). Convolutional neural network for semiconductor wafer defect detection. *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. <https://doi.org/10.1109/icccnt45670.2019.8944584>
- Gao, H., Zhang, Y., Lv, W., Yin, J., Qasim, T., & Wang, D. (2022). A deep convolutional generative adversarial networks-based method for defect detection in small

sample industrial parts images. *Applied Sciences*, 12(13), 6569.
<https://doi.org/10.3390/app12136569>

Ghosh, K., Bellinger, C., Corizzo, R., Branco, P., Krawczyk, B., & Japkowicz, N. (2022). The class imbalance problem in deep learning. *Machine Learning*.
<https://doi.org/10.1007/s10994-022-06268-8>

Hsu, C.-Y., & Chien, J.-C. (2020). Ensemble Convolutional Neural Networks with weighted majority for Wafer bin Map Pattern Classification. *Journal of Intelligent Manufacturing*, 33(3), 831–844. <https://doi.org/10.1007/s10845-020-01687-7>

Hyun, Y., & Kim, H. (2020). Memory-augmented convolutional neural networks with triplet loss for imbalanced wafer defect pattern classification. *IEEE Transactions on Semiconductor Manufacturing*, 33(4), 622–634.
<https://doi.org/10.1109/tsm.2020.3010984>

Ji, Y., & Lee, J.-H. (2020). Using GAN to improve CNN performance of Wafer map defect type classification: Yield Enhancement. *2020 31st Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*.
<https://doi.org/10.1109/asmc49169.2020.9185193>

Kong, Y., & Ni, D. (2019). Recognition and location of mixed-type patterns in wafer bin maps. *2019 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)*.
<https://doi.org/10.1109/smile45626.2019.8965309>

Kong, Y., & Ni, D. (2020). Qualitative and quantitative analysis of multi-pattern wafer Bin Maps. *IEEE Transactions on Semiconductor Manufacturing*, 33(4), 578–586.
<https://doi.org/10.1109/tsm.2020.3022431>

Kumar, N. S., Mahato, R. K., & Jana, Dr. S. (2023). Semiconductor Wafer Defects Detection using Yolov5 Model. *International Journal of Novel Research and Development*, 8(10), b601–b606.
<https://doi.org/https://www.ijnrd.org/papers/IJNRD2310164.pdf>

Kyeong, K., & Kim, H. (2018). Classification of mixed-type defect patterns in wafer bin maps using convolutional Neural Networks. *IEEE Transactions on Semiconductor Manufacturing*, 31(3), 395–402.
<https://doi.org/10.1109/tsm.2018.2841416>

Lee, H., & Kim, H. (2020). Semi-supervised multi-label learning for classification of Wafer bin maps with mixed-type defect patterns. *IEEE Transactions on Semiconductor Manufacturing*, 33(4), 653–662.
<https://doi.org/10.1109/tsm.2020.3027431>

- Li, X., Duan, C., Zhi, Y., & Yin, P. (2021). Wafer crack detection based on Yolov4 Target Detection Method. *Journal of Physics: Conference Series*, 1802(2), 022101. <https://doi.org/10.1088/1742-6596/1802/2/022101>
- MIR Corpora. (2015). <http://mirlab.org/dataSet/public/>
- Nag, S., Makwana, D., R, S. C., Mittal, S., & Mohan, C. K. (2022). WaferSegClassNet - a light-weight network for classification and segmentation of semiconductor wafer defects. *Computers in Industry*, 142, 103720. <https://doi.org/10.1016/j.compind.2022.103720>
- Park, S., & You, C. (2023). Deep convolutional Generative adversarial Networks-Based data augmentation Method for classifying Class-Imbalanced defect patterns in Wafer BIN MAP. *Applied Sciences*, 13(9), 5507. <https://doi.org/10.3390/app13095507>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. <https://doi.org/10.1109/cvpr.2016.91>
- Ren, S., He, K., Girshick, R., & Sun, J. (2017b). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149. <https://doi.org/10.1109/tpami.2016.2577031>
- Saqlain, M., Abbas, Q., & Lee, J. Y. (2020). A deep convolutional neural network for wafer defect identification on an imbalanced dataset in Semiconductor Manufacturing Processes. *IEEE Transactions on Semiconductor Manufacturing*, 33(3), 436–444. <https://doi.org/10.1109/tsm.2020.2994357>
- Shin, W., Kahng, H., & Kim, S. B. (2022). Mixup-based classification of mixed-type defect patterns in Wafer Bin Maps. *Computers & Industrial Engineering*, 167, 107996. <https://doi.org/10.1016/j.cie.2022.107996>
- Shinde, P. P., Pai, P. P., & Adiga, S. P. (2022). Wafer defect localization and classification using Deep Learning Techniques. *IEEE Access*, 10, 39969–39974. <https://doi.org/10.1109/access.2022.3166512>
- Tello, G., Al-Jarrah, O. Y., Yoo, P. D., Al-Hammadi, Y., Muhaidat, S., & Lee, U. (2018). Deep-structured machine learning model for the recognition of mixed-defect patterns in semiconductor fabrication processes. *IEEE Transactions on Semiconductor Manufacturing*, 31(2), 315–322. <https://doi.org/10.1109/tsm.2018.2825482>
- Tsai, D.-M., Huang, Y.-Q., & Chiu, W.-Y. (2021). Deep learning from imbalanced data for automatic defect detection in Multicrystalline Solar Wafer images.

Measurement Science and Technology, 32(12), 124003.
<https://doi.org/10.1088/1361-6501/ac1fbf>

Ultralytics. (2024, May 18). *Classify*. Classify - Ultralytics YOLOv8 Docs.
<https://docs.ultralytics.com/tasks/classify/>

Wang, J. (2020). WaferMap Dataset: MixedWM38.
<https://github.com/Junliangwangdhu/WaferMap>

Wang, J., Xu, C., Yang, Z., Zhang, J., & Li, X. (2020). Deformable convolutional networks for efficient mixed-type wafer defect pattern recognition. *IEEE Transactions on Semiconductor Manufacturing*, 33(4), 587–596.
<https://doi.org/10.1109/tsm.2020.3020985>

Wang, J., Yang, Z., Zhang, J., Zhang, Q., & Chien, W.-T. K. (2019). Adabalgan: An improved generative adversarial network with imbalanced learning for Wafer Defective Pattern recognition. *IEEE Transactions on Semiconductor Manufacturing*, 32(3), 310–319. <https://doi.org/10.1109/tsm.2019.2925361>

Wei, Y., & Wang, H. (2022). Mixed-type wafer defect pattern recognition framework based on multifaceted dynamic convolution. *IEEE Transactions on Instrumentation and Measurement*, 71, 1–11.
<https://doi.org/10.1109/tim.2022.3178498>

Yamashita, R., Nishio, M., Do, R. K., & Togashi, K. (2018). Convolutional Neural Networks: An overview and application in Radiology. *Insights into Imaging*, 9(4), 611–629. <https://doi.org/10.1007/s13244-018-0639-9>

Yang, W., Peng, J., Du, J., Sheng, L., Lu, X., & He, Z. (2024). Wafer Defect Detection Technology Based on CTM-Iyolov5 Network.
<https://doi.org/10.1364/opticaopen.24916026>

Yu, N., Chen, H., Xu, Q., Hasan, M. M., & Sie, O. (2022). Wafer map defect patterns classification based on a lightweight network and data augmentation. *CAAI Transactions on Intelligence Technology*. <https://doi.org/10.1049/cit2.12126>

Zhai, X., Huang, Z., Li, T., Liu, H., & Wang, S. (2023). Yolo-drone: An optimized YOLOv8 Network for tiny UAV object detection. *Electronics*, 12(17), 3664.
<https://doi.org/10.3390/electronics12173664>

Zhou, N., Liu, Z., & Zhou, J. (2022). YOLOv5-based defect detection for Wafer Surface Micropipe. *2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*.
<https://doi.org/10.1109/ispds56360.2022.9874083>