Deep Transfer Learning for Macroscale Defect Detection in Semiconductor Manufacturing

by

John Timothy Waterworth

B.S. Mechanical Engineering

Texas A&M University, 2022

Submitted to the Department of Mechanical Engineering in partial fulfillment
of the requirements for the degree of

MASTER OF ENGINEERING IN MECHANICAL ENGINEERING
in conjunction with the Advanced Manufacturing and Design Innovation Program

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by: John T. Waterworth
Department of Mechanical Engineering
August 11, 2023

Certified by: Duane Boning
Clarence J. LeBel Professor, Electrical Engineering and Computer Science
Thesis Supervisor

Certified by: David E. Hardt
Professor of Mechanical Engineering; Ralph E. and Eloise F. Cross Professor in Manufacturing
Thesis Supervisor

Accepted by: Nicolas Hadjiconstantinou
Professor of Mechanical Engineering
Graduate Officer, Department of Mechanical Engineering

PAGE INTENTIONALLY LEFT BLANK

Deep Transfer Learning for Macroscale Defect Detection in Semiconductor Manufacturing

by

John Timothy Waterworth

Submitted to the Department of Mechanical Engineering on August 11, 2023, in partial fulfillment of the requirements for the degrees of

Master of Engineering in Mechanical Engineering

Abstract

This thesis proposes improvements to wafer macro inspection processes and tools on four axes at Texas Instruments. The major axis of improvement involves real-time machine learning recommendations regarding the presence of macroscale defects. In this work, a model for detecting central defects is described in detail, and a novel approach to overcoming data scarcity through the creation of synthetic data is deployed. The binary classifier model achieves an out-of-distribution area under curve (AUC) of 0.909 for detecting hotspot defects. Detection for other classes of central defects is also explored but limited by even greater data sparsity. Models for catching spin on glass defects and edge defects are also trained with out-of-distribution AUCs reaching 0.927 and 0.906 respectively. Other axes of improvement covered in this thesis involve gauge reliability and repeatability analysis of macro inspection tools, the creation of a new user interface called OwlView, and the trial of a new macro inspection system used in-line on photolithography tools for greater efficiency. Gauge repeatability and reliability analysis gives insight into tool function and assists the team and technicians in root cause analysis. Several hardware failures of current toolsets are identified and addressed. Maintenance procedures are also updated to keep tools operating within specifications. The OwlView interface is developed with features to increase user efficiency. Additionally, the interface helps create an infrastructure for tagging more data, which will be fed back into the models to address data scarcity. Lastly, an in-line inspection trial shows achievable high quality wafer images compatible with the machine learning and inspection infrastructures developed in this work.

Thesis Supervisor: Duane Boning

Title: Clarence J. LeBel Professor, Electrical Engineering and Computer Science

PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

Thank you to the Advanced Manufacturing and Design Innovation program at MIT and Texas Instruments for making this collaboration happen. Specifically, I thank our project advisor Duane Boning, program director Jose Pacheco, and Professor Dave Hardt at MIT. I am also very grateful to our project sponsors Todd Binnix and Vidya Krishnan at Texas Instruments. Additionally, there are countless engineers, technicians, specialists, managers, and so forth we worked with this summer I would like to thank. Most importantly, thank you to Jonathan and Sophia for coming to Texas and completing this project with me. Finally, thank you to my parents, brother, and friends for their endless support.

PAGE INTENTIONALLY LEFT BLANK

Contents

PAGE INTENTIONALLY LEFT BLANK

List of Figures

List of Tables

PAGE INTENTIONALLY LEFT BLANK

Glossary

Architecture - A machine learning model's arrangement and sizing of layers and functions.

Class Imbalance - Varying sample sizes between classes fed into a machine learning model.

Convolutional Neural Network (CNN) - A type of artificial neural network commonly used for image recognition that uses mathematical convolutions in one or more of the layers.

Defect - An unwanted feature imparted to a wafer that will lead to scrap or yield loss.

Die - An individual circuit pattern that exists within a wafer before cutting and packaging.

Fab - A facility where wafers containing die are produced.

Gallium Nitride (GaN) - A composite semiconductor material with a higher electron mobility, operational temperature limit, and power density than silicon.

Hyperparameter - A machine learning parameter that controls its learning process and is set before training begins.

Layer / Mask Level - The layer of the wafer in the manufacturing process created through photolithography which dictates part geometry.

Logpoint - A four-digit identifier number corresponding to the layer level and processes a lot of wafers have undergone to a certain point.

Lot - A group of 25 wafers held together in a cassette.

Macroscale Defects - Defects that are visible without needed magnification.

Poka-Yoke - A measure that prevents human error in a manufacturing environment.
ROC (Receiver Operating Characteristic) Curve - A 2D plot showing a binary classifier's capabilities on a given data set.

Root Cause Analysis (RCA) - A process in which the cause of systematic variation in product or process quality is diagnosed.

Scrap - A wafer that contains so many defects that it is thrown out entirely, meaning 100% of the die are lost.

SOG - Refers to a process called "Spin on Glass" in which an insulator layer is added after metal levels.

SOG Defects - Non-reworkable defects arising from the SOG process in which a solvent splatters all over a wafer.

Transfer Learning - The process of using initial machine learning model weights that are the result of a different and often larger data set.

Wafer - A circular disc of crystalline semiconductor material that circuitry is built upon.

Yield Loss - The loss of die on a wafer attributable to some random or systematic variation in the manufacturing process.

Chapter 1

Introduction

In recent years, the semiconductor industry has seen - and is undergoing - unprecedented growth. This growth is driven by the exponentially increasing demand for chips in products like vehicles and smart devices [1]. Paired with this growth is a newfound incentive to produce more chips domestically given shortcomings in supply chains exposed by the COVID pandemic and geopolitical tensions between China and Taiwan [2]. It is crucial that wafer fabs can produce as many chips as possible to meet this demand. While newer 300mm fabs are being built to accommodate this, older 200mm wafer fabs still must and do produce chips. Although more advanced in age, these fabs produce cutting-edge technologies like Gallium Nitride (GaN) wafers. Unfortunately, the fabs are not equipped with the most advanced systems and are often highly manual regarding tool loading and wafer inspection. This thesis details the development and application of software, procedures, and machine learning (ML) models used for defect detection in the semiconductor industry's older fabs. In this chapter, the background and motivation for the work is detailed. Additionally, items such as a problem statement and thesis structure are described.

1.1 Project Motivation

Macroscale wafer inspection occurs within the photolithography module at Texas Instruments's (TI's) 150mm and 200mm fab, DFAB. Inspection involves the wafers being sent through an EAGLEView (EV) macro inspection tool that images the wafers and marks them for defects. All images are then reviewed by an operator who sends wafers that look defective to microscopic inspection, and then they are reworked if a defect is confirmed. It is crucial that defects are caught within this module since they are reworkable. If they are missed and sent to another module, they will be turned to scrap. The current inspection process is flawed on several fronts. First, the inspection tool used to image and flag defective wafers operates on a rules-based algorithm with a high miss rate. Additionally, seven different models of this tool are in use, and their performance is not repeatable or reliable - as exemplified later in Chapter 3. Further, operators review every single wafer image and make the ultimate decision to flag a wafer in the end. This is problematic as human judgment is highly variable, and it is impossible to review thousands of wafer images daily at a high and consistent standard. Lastly, the software that the operator utilizes to review the images lacks features or filtering options to view defects efficiently.

As a result of these flaws, TI has sponsored our work to revamp their macroscale inspection capabilities with ML models. The scope of our work can be defined under four axes of improvement. First, to understand and quantify current tool and operator performance. Second, to make the macro inspection process more effective by creating a new graphical user interface (GUI) for operators to inspect wafers. Third, to create ML models that identify defects and can be integrated into the GUI developed. And fourth, to identify and pilot new inspection methods that are in-line with 200mm tools and compatible with the new GUI and ML frameworks.

1.2 Problem Statement

Before this work, no significant projects had been pursued at DFAB to revamp the macro inspection process. Quotes for new EAGLEView tools have been gathered, but with a price tag over $500,000 per tool, there is no justifiable return on investment (ROI) for adoption. Additionally, other less expensive macro inspection tools are typically incompatible with DFAB given its 200mm production size. With the goal of catching as many macroscale defects as possible, several hypotheses are generated. First, the massive repository image scans from the EAGLEView tools can be utilized to create ML algorithms for different defect classes. This proves to be a significant undertaking, as while large amounts of data are available, large amounts of sorted and tagged data are not. Additionally, there are class imbalance issues given that different defect types occur at different rates across technologies and layer levels. Further, image scans of wafers come with some tool-dependent artifacts that manifest as noise ML models may wrongly cling to.

A second hypothesis is that superior, in-line macro inspection is viable for DFAB, and can be implemented to antiquate the problematic EAGLEView toolsets at a justifiable ROI. To test this hypothesis, a trial imaging system is mounted on a spin coater tool, evaluated, and proven to be compatible with the GUI and ML framework developed through the scope of this work.

1.3 Approach

Given the broad scope of work covered in this study, a high degree of interdisciplinarity, collaboration, adaptability, and analysis is required. Early on, a thorough understanding of the macro inspection process is garnered through GR&R analysis and time on the production floor. The new operator GUI is developed continually with input from manufacturing management, operators, IT, and engineering. New in-line wafer imaging options are explored and a model is selected and trialed.

In parallel, image data for ML models is gathered, tagged, and sorted utilizing custom engineering tools developed by our team. One such tool is a modified version of the new GUI.

Convolutional neural network (CNN) models are developed using VGG-11 with pre-trained weights and biases to identify specific high-priority defects that arise from spin-on-glass (SOG) tools. Additional models for edge and central defects are also developed. The models utilize data augmentation and normalization techniques to address limited data and the wide variance of defect sizes. Issues regarding implementing these models to the GUI, such as inference time, are also addressed. A framework is also established such that more models can be devised and implemented with our system in the future. This is true whether models are trained on the EAGLEView data set, or new image types from in-line inspection.

1.4 Division of Work

In addressing the wide array of work conducted throughout the summer, Sophia Cheung, Jonathan Sampson, and I all contributed to each of the axes of improvement in significant ways. The axes of improvement often intertwine and lend key insight into how the other elements of our work take effect. For the scope of this thesis, the development of the GUI, the creation of an ML infrastructure, and innovations regarding the ML model for center defects are covered in-depth. Relatively short summaries of the other sections are presented in this work, and more in-depth descriptions are found in the theses of Sophia Cheung [35] and Jonathan Sampson [6].

1.5 Thesis Outline

Chapter 2 details the manufacturing equipment and procedures related to DFAB's macro inspection module, along with improvement goals for each of the axes. The work done in regard to GR&R and critical findings is summarized in Chapter 3. Chapter 4 outlines the design and implementation of a GUI for image inspection, as well as its use as an ML data tagging tool. Chapter 5 covers the availability of data and the development of additional tools to prepare data for ML training. Next, Chapter 6 introduces macro-scale defect detection in the semiconductor industry. Chapter 7 introduces and discusses different machine learning methodologies and concepts applied to our detection algorithms. The use of our ML infrastructure to train a model for detecting spin-on-glass (SOG) defects is summarized in Chapter 8. Chapter 9 summarizes the use of our ML infrastructure to train a model for recognizing edge defects. Next, Chapter 10 gives an in-depth analysis of methods used to train an ML model to recognize center defects. Chapter 11 summarizes the team's findings surrounding the use of in-line inspection at DFAB. Chapter 12 presents conclusions and future work relating to the macro inspection module at DFAB.

Chapter 2

Problem Background

This chapter outlines background information relevant to semiconductor manufacturing and wafer macro inspection at DFAB. The overall manufacturing and inspection process is first overviewed, followed by a discussion of the improvements desired in the macro inspection process. The key points of the chapter are then summarized.

2.1 Manufacturing and Inspection Overview

This section first introduces the overall existing fab and manufacturing process at TI. The macro inspection process is then described in more detail, which is the focus of this thesis.

2.1.1 Overview of Wafer Manufacturing at DFAB

One of TI's oldest wafer fabs is DFAB, which began operation under that name in the late 1990s - although it ran its first production wafer in the late 1960s. DFAB is unique in that while being among TI's oldest fabs, it manufactures some of its most innovative products like GaN wafers - which yield chips with higher power densities and switching speeds due to the substrate's elevated electron mobility [3]. DFAB produces both 150mm and 200mm wafers, with 150mm production phasing out over the next few years. It has found its niche within TI by both absorbing technologies produced by other older fabs as they close down and by pioneering the production of GaN technologies on smaller wafer sizes - since GaN cannot be scaled to 300mm production yet [4]. The vision for DFAB is to become a 200mm GaN fab. Consequently, it is undergoing a massive scale-up to meet production goals among many changes. Contrary to newer 300mm fabs, DFAB is highly manual, and tools are typically not covered by the manufacturer. As a result, a high degree of innovation is required to keep DFAB competitive with other more advanced fabs.

Typically, toolsets within fabs are supported by tool vendors, but outside of the GaN and Epitaxial Film modules, this is rarely the case at DFAB due to its advanced age. As a result, management is set to meet ambitious production goals for GaN with older and problematic toolsets. Additionally, it is difficult to replace these toolsets since providers have shifted focus to 300mm technologies. Thus, a high degree of innovation is required to keep DFAB running and to meet the growing demand for GaN technologies.

Like other wafer fabs, DFAB is composed of many modules that perform specific operations on the wafers. Macro inspection happens just after the photolithography module. This is key as defects generated by photolithography are reworkable. If reworkable defects are missed, wafers travel to other modules where nonreversible processes are carried out, and scrap or yield loss is generated. Figure 2-1 shows an outline of material flow through a typical wafer fab.

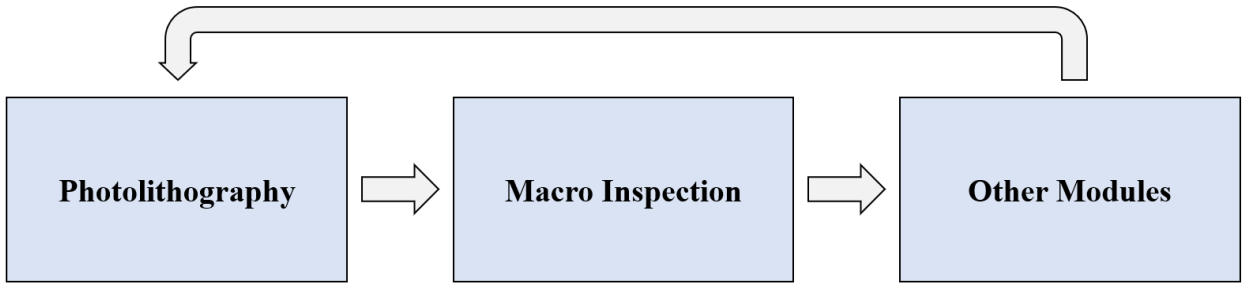


Figure 2-1: General flow of wafers through a fab. Wafers are coated in a photocurable resin in the photolithography module and then selectively cured to create geometric features. These layers are subsequently macro-inspected on EAGLEView tools. In other modules, various permanent processes are conducted.

A crucial metric for overall fab efficiency is days per mask level (DPL). This is the average time it takes for a wafer to return to the photolithography module after its previous layer. Upper management aims to drive this number down so that chips can be delivered to customers faster.

2.1.2 Macro Inspection Process Overview

DFAB macro inspects wafers using seven different EAGLEView (EV) tools that flag wafers for defects after the photolithography steps. Manufacturing specialists load wafers onto the EAGLEView tools, which scan the wafers individually. Next, these images are flagged by EV's rules-based algorithms and sent to a server. From here, manufacturing specialists review every single image and EAGLEView decision and then make a judgment on whether or not to flag each wafer for a microscopic inspection. Figure 2-2 outlines the current process flow for macro inspection.

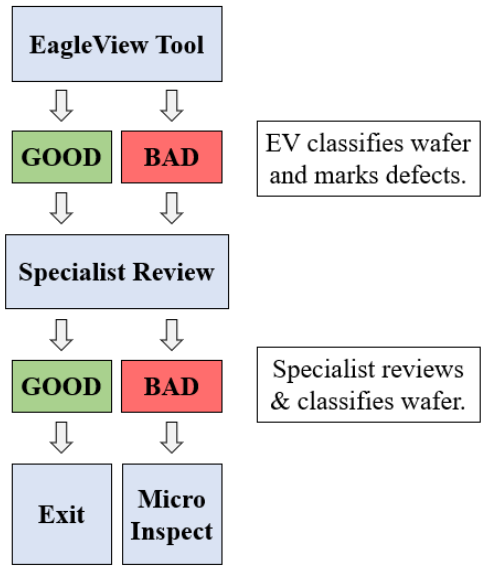


Figure 2-2: Inspection process flow. EAGLEView makes a classification decision and marks wafer scans as good or bad. These scans are sent to the specialist to review on a desktop PC next to EAGLEView. The specialists decide whether or not the wafer scan is truly good or bad via visual inspection of the image.

Within this system, there are two key judgments made on a wafer that dictate whether or not it is flagged for microscopic inspection. The first is the decision made by the EAGLEView tool the wafer is run on to flag the wafer as good or bad. The EAGLEView uses a rules-based golden wafer algorithm that looks for discrepancies between the first wafer scanned and the rest of the lot. While powerful at times, there are numerous performance issues. In short, the performance of this algorithm is highly variable based on both the wafer mask level and the tool's hardware, as discussed in Chapter 3. Wafer mask levels lead any given wafer to have a wildly different appearance throughout its journey as displayed by Figure 2-3, complicating adjustments necessary for EAGLEView software to be effective - particularly for dark logpoints.

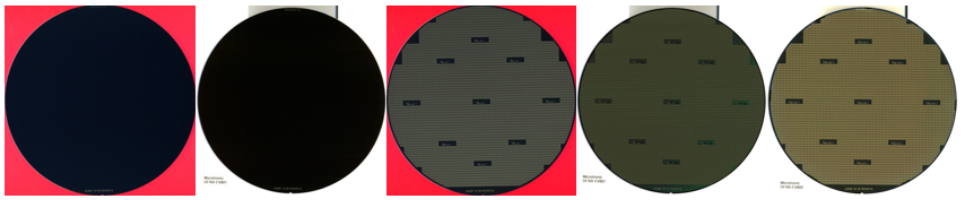


Figure 2-3: Wafer scans at varying logpoints for a single wafer. The same wafer has varying optical properties at each logpoint, and EAGLEView's algorithm struggles with classifying darker layers. At each scan, the background color and presence of artifacts also vary with the EAGLEView the lot is run on.

The second key judgment is made by the manufacturing specialist on shift. This specialist is often responsible for both loading EAGLEView tools and reviewing the wafer images. Shifts run three to four days a week for twelve hours every day, and the module is typically staffed with two specialists at a time. Specialists utilize software developed by EAGLEView for review that is riddled with issues. It randomly quits, runs slow when multiple operators use it, allows specialists to submit lots without reviewing them, has no color equalization filtering for darker layers, and so on. As a result, specialists are not as effective as they can be at catching defects. The specialists' decisions are governed by a detailed standard operating procedures (SOP) document with details on over 80 different defect classes they are expected to identify. Further, there are details regarding the shape, size, and number of defects tolerable for a given wafer or lot. Table 2-1 shows the possible specialist decisions and associated outcomes of each.

Table 2-1: Possible specialist decisions for good and bad wafers.

Reality:	**Good**		**Bad**	
Specialist Decision:	Good	Bad	Good	Bad
Case:	True Negative	False Positive	False Negative	True Positive
Outcome:	Wafer Continues	Wasted Time	Defect Missed	Defect Caught

In the above table, a comparison between the specialist decision and reality is made, and the associated outcomes are identified. If the wafer is good, and the specialist agrees, the wafer exits the module with no hold for micro inspection. If the wafer is good but the specialist thinks it is bad, it is wrongly sent to micro inspection - wasting time and resources. If the wafer is bad but the specialist thinks it is good, the defect is sent through the module and the defect will be turned into permanent yield loss or scrap. Yield loss refers to cases where localized parts of the wafer are affected and it is still run through

production. Scrap refers to cases where the whole wafer must be thrown out. Both are very costly. The final case is the wafer being bad, and the specialist agreeing. In this case, the defect is caught at micro inspection and then the wafer is reworked - preventing scrap or yield loss.

Performance by the specialist is variable, as team leads have years of reviewing experience whereas newer hires have proven harder to train and retain. Typically, specialists err on the side of caution, flagging a far higher number of wafers than ones that have defects. In fact, the average monthly flagging rate is 35%. In contrast, the average rework rate is 0.2%, indicating 34.8% (35% - .2%) of wafers are flagged and sent to micro inspection without being identified as needing rework by the micro inspection process. Micro inspection is much slower than macro inspection, so the micro inspection station is typically staffed with three specialists. If the false positive rate of the EAGLEView and specialist review can be reduced, the number of specialists required to inspect at a given time can be as well, impacting operational costs.

Conversely, specialists also miss defects. When wafers with critical defects pass through EV, they are turned to scrap. On average, 28 wafers/month are identified as passing through EAGLEView with a reworkable defect and then turning into scrap. Additionally, there are numerous instances of chip yield loss associated with smaller defects not being identified by EV. Lastly, defects from other modules have the potential of being caught at EV. If these instances of false negatives by the combined EAGLEView and specialist review can be reduced, significant material savings can be generated.

2.2 Improvement Axes and Goals

Our team identified four main axes of improvement for DFAB’s macro inspection process: GR&R, enhanced specialist software and procedures, ML-aided algorithms, and an in-line inspection investigation. The goals associated with each project section are described in the subsections below.

2.2.1 GR&R Goals

Systemic issues exist within the EAGLEView toolset and the procedures surrounding them. To address this, the team plans to conduct a GR&R analysis to help identify root causes. In doing so, an immediate impact on the module's performance is had. Subsequently, based on findings, procedures and recommendations are generated to prevent systematic shifts in tool performance in the future.

2.2.2 Improved User Interface Goals

Manufacturing specialists can be made more effective through a new GUI. The current review software specialists use is flawed on several fronts that hinder operator performance. The team aims to build a new GUI based on the needs described by specialists, engineers, and managers. Of these, features like enhanced filtering, speed improvements, and an inspection tracking mechanism are paramount. This avenue is also pursued since, to give specialists real-time ML-based recommendations, an interface is required.

2.2.3 Machine Learning Goals

Within the macro inspection module, wafer scans go through two layers of critique for flagging: the EAGLEView algorithm and the specialist's visual review on the EAGLEView software. Each of these has its own suite of issues. The EAGLEView algorithm is rules-based, and relies upon a golden wafer technique in which the first wafer scanned is assumed perfect and the rest of the lot is compared to it. As a result, when defects affect a whole lot, or there is an issue with the golden wafer, inaccuracies grow. Additionally, there are performance issues within and between tools that lead to poor accuracy. Ultimately, the decision is made by the specialist whether or not to flag a wafer for micro inspection. Unfortunately, specialists must review thousands of images over long shifts leading to high degrees of variability and subjectivity.

To address these issues, a third layer of critique is added of ML algorithms for defect detection. In theory, ML is superior to rules-based algorithms at identifying different defects across a wider array of wafer technologies since they can abstract features. Additionally, there is no reliance on a golden wafer, so defects that affect the whole lot will not be missed as they are by EV.

DFAB had no infrastructure for creating or training ML algorithms at the start of our work. While true, DFAB does have a large archive containing scans of hundreds of thousands of wafers from the EAGLEView module. So, the goal of the first stage is to create a robust infrastructure for ML at DFAB. Next, it is to train several ML models that target the most persistent and costly defects slipping through the macro inspection module. The first of these are SOG defects, among the smallest and hardest to recognize for EVs and specialists. The next are edge defects, which are commonly the result of an edge bead removal (EBR) or photoresist dispense that is out of control. The final is for defects affecting the center of the wafers, which are commonly manifest in areas of photoresist not cured properly due to a slew of different root causes.

2.2.4 In-line Inspection Pilot Goals

Increasingly, fabs have shifted to in-line macro inspection of wafers because it can help reduce the DPL of a fab [5]. Due to this, and the fact that DFAB is shifting to a 200mm only GaN fab over the next few years, inspection of wafers in-line on Tokyo Electronic (TEL) tools is explored. For the pilot, the team aims to select an imaging method, design mounts, install a trial unit on a tool, validate image quality, validate image data flow, prove integrability to our GUI and ML infrastructure, and create a detailed financial model and recommendation for upper management based upon findings.

2.3 Chapter Summary

Within this chapter, a brief overview of DFAB and semiconductor manufacturing was provided. The flow of wafers through the macro inspection module was detailed. The EAGLEView rules-based golden wafer algorithm was introduced, as well as details surrounding the human inspection. Key needs were identified based on the current system as being GR&R, a new GUI, ML-based detection, and exploration of in-line inspection.

Chapter 3

Gauge Repeatability and Reproducibility Analysis

In this chapter, a high-level overview of the GR&R study conducted on the macro inspection toolset is covered. For a more detailed analysis of the work conducted by the team, please refer to Sampson [6].

3.1 Summary of GR&R Purpose and Procedure

A GR&R study is conducted to evaluate EAGLEView performance with respect to three primary questions. These are as follows

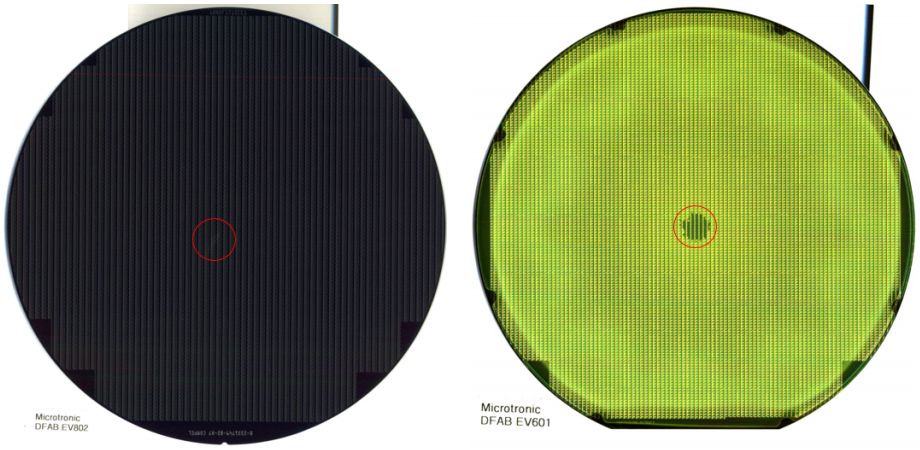
- Can an EAGLEView tool accurately locate a known defect on a given wafer and correctly classify
  the wafer?
- Will a single EAGLEView tool output consistent classifications and defect localization markings each time a wafer is run through the tool?
- Do all EAGLEView tools provide the same classifications and defect localization markings for a given wafer?

To understand these ends, an experiment is devised for both the 200mm and 150mm EAGLEView toolsets. This experiment is simplistic and brief so that the production floor is minimally disrupted. Lots containing one to five known defective wafers are run through each toolset multiple times. This is conducted with wafer randomization deactivated and other tool settings standardized. This is key so that each EAGLEView has the same golden wafer between runs. Due to time and production limitations, this procedure is only done once for each wafer size. Table 3-1 below shows the evaluation criteria utilized to answer each of the three questions posed by the team.

Table 3-1: Questions and evaluation criteria for GR&R study [6].

Investigation Question	Evaluation
Can an EAGLEView tool accurately locate a known defect on a given wafer and correctly classify the wafer?	- qualitative evaluation of defect localization accuracy - binary wafer classification accuracy
Will a single EAGLEView tool output consistent classifications and defect localization markings each time a wafer is run through the tool?	- inter-trial comparison of classification and defect localization results - comparison of accuracy metrics and confusion matrices - qualitative comparison of defect localization
Do all EAGLEView tools provide the same classifications and defect localization markings for a given wafer?	- binary wafer classification accuracy and comparison across tools - qualitative comparison of defect localization across tools

For the studies conducted, manufacturing specialists identified and ran lots for the 200mm and 150mm tools. The defective wafers utilized for both studies are shown below in Figure 3-1.



200mm Defect 150mm Defect

Figure 3-1: Defective wafers for GR&R study. The left image shows the 200mm wafer with a known center defect as utilized in the 200mm GR&R trial. The right image shows the 150mm wafer with a known center defect as utilized in the 150mm GR&R trial.

As discussed in the following paragraphs, a more thorough evaluation would involve many defect types across many different layers and technologies. While advantageous, the team is limited to defective

lots identified by specialists and short timeframes to keep production running. A test matrix is shown in Table 3-2 below.

Table 3-2: GR&R testing matrix [6].

Input				Output
Lot Size	**Wafer Size**	**Known Defects**	**Scan Trials**	
25	200mm	#15	2+	EV Classifications: [GOOD]/[BAD]
25	150mm	#6, #8, #9, #15, #22	2+	EV Localizations: Annotations in Red

For the 200mm study, a lot containing one known defect is utilized. For the 150mm study, a lot with five known defects is utilized. For both, the EV's classification and ability to localize the defect is evaluated.

3.2 Summary of GR&R Results and Corrective Action

The study results show that the 200mm EAGLEView tools perform poorly in repeatability and reliability, whereas the 150mm tools display better results. Figure 3-2 shows the EAGLEView classifications for the 200mm toolsets, whereas Figure 3-3 shows the classifications for 150mm toolsets.

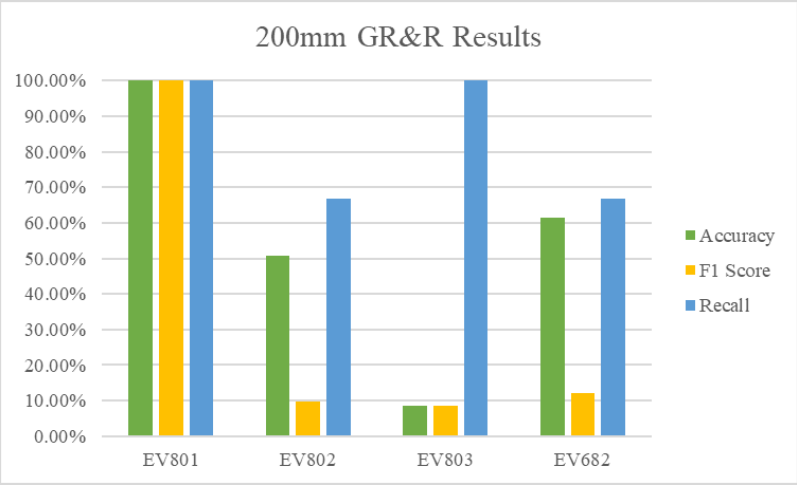


Figure 3-2: EAGLEView classifications for 200mm GR&R [6]. Results for each EAGLEView tool are averaged across three trials for all except EV803 (2 trials).

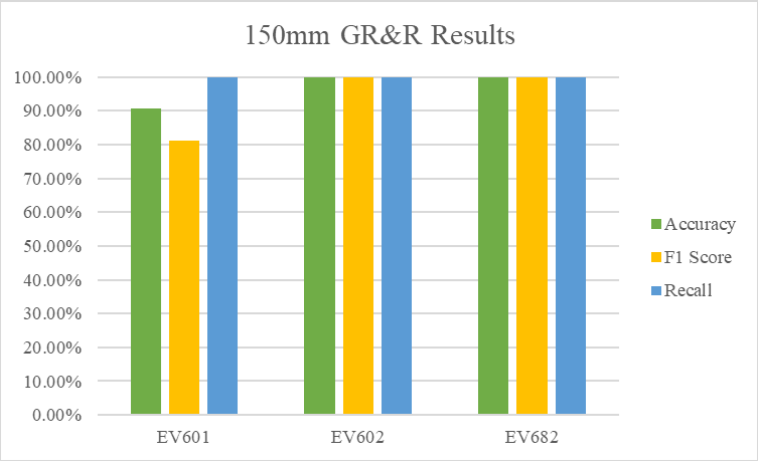


Figure 3-3: EAGLEView classifications for 150mm GR&R [6]. Results for each EAGLEView tool are averaged across three trials.

The 200mm tools perform considerably worse than the 150mm tools in this study, which may be attributed to the small defect size utilized in the 200mm trial and the dark layer level. Regardless, EV801 still identifies and localizes the defect in each of the three trials performed on it, indicating it was detectable for EV's algorithms. Because the EAGLEView software and settings are baselined across tools, root cause analysis is initiated for the hardware problematic EVs. Table 3-3 shows the actions taken and findings for root cause analysis (RCA) conducted by EAGLEView technicians for EV802.

Table 3-3: Root cause analysis actions and findings for EV802 [6].

Action Taken	Findings
Clean scanner, replace scanner cathode fluorescent lamp and neutral density filter	Issue unresolved, all components in working order
Evaluate internal component layout	All components tightly packed together, not properly mounted–resting on other components. Power supply showing signs of significant temperature rise, potentially affecting proximate components
Evaluate power supply	Power supply showing signs of failure
Evaluate CPU Control board	Adhesive residue from tape melting and dripping onto CPU Control board due to internal temperature rise
Replace power supply	**Classification and localization results improved**

Investigation conducted by a technician reveals that the power supply was overheating, leading to random noise and false flagging of defects across the wafers (thus the high recall percentages). Additionally, the investigation spurred the creation of new mounting hardware and components to prevent issues with overheating and warping of the neutral density filters on some EAGLEView tools. As a result of these findings, the team recommends the following preventive maintenance proceedings be conducted.

Diagnostic Testing

Recurring GR&R Study

- Perform a GR&R study similar to the investigation designed in this work every two months
- Evaluate all EAGLEView tools, and replicate the study for a variety of layers and logpoints
- Use results to determine the frequency of maintenance activity

Tool Evaluation and Maintenance

As a result of the recurring GR&R study, the following evaluation procedures are recommended if the investigation yields results indicating the tools require service.

Neutral Density Filter

- Check filter and glass cover status
    - ensure the filter is not warping due to temperature rise

Power Supply

- Check power supply status and replace if necessary

Scanner Light Source

- Check lamp status, replace scanner if necessary

Scanner

- Evaluate dust and particle accumulation on scanner head

Scanner Guide Rod

Guide rod already lubricated every month

3.3 Summary of GR&R Future Work

The investigation described in this chapter is rather limited in that it only covers one defective lot for each of the wafer sizes. The introduction of different defect types across different layer levels would help make the study more robust and potentially lead to more discoveries of hardware failures on each of the tools. Along with this, an investigation into how the EVs perform given differing tool settings would be useful. Perhaps with different thresholds and resolutions, the tools will perform better for certain defect classes.

Chapter 4

Graphical User Interface Development

This chapter details the new graphical user interface developed for wafer inspection. The interface is designed for use by specialists on the production floor and by engineers in the collection and tagging of data. Its motivation, design, and implementation are covered in this chapter.

4.1 Motivation for New User Interface

Development of a new user interface is motivated by three key factors. First, to present real-time ML-based recommendations, an interface that is easy to interpret for a manufacturing specialist is required. The EAGLEView software utilized by manufacturing exists in the form of a compiled executable with no editable source code. As a result, it is impossible to integrate ML models into the current display. Without a new GUI, two interfaces would have to be used by the operator in parallel, which is not a robust or practical solution.

Second, the current inspection software is riddled with performance issues that are detrimental to module performance. The initial issue described by engineering management to the team is a lack of wafer image filtering on the production floor. Wafer defects exist in various shapes and sizes, across a wide array of layer types. Certain layer and defect types are nearly impossible to discern without filtering, hindering operator performance. The only filtering option in the current production software is a simple color inversion, which has not proven helpful in defect identification. Management and engineering also expressed the desire for specialists to be able to review the same wafer scans as they existed at previous log points to observe how the defect has progressed between layers. To achieve this in the current software, specialists are required to reload each individual logpoint. With each change of logpoints taking up to minutes, this is often too time expensive to execute on the production floor. As a result, specialists have no practical insight into how a defect is progressing between layers on the production floor. The final key issue expressed by management and engineers is the lack of specialist tracking and poka-yokes. In the current software, specialists can pass a whole lot of wafers without actually reviewing a wafer image. Further, engineering and management have no insight into how long each wafer image was reviewed by the specialist on shift. Consequently, engineering often has difficulties identifying why or how a specialist decided to pass a wafer.

Finally, the development of a custom GUI is motivated by the fact it can be repurposed in several useful applications. First, it can be utilized to tag data for ML models more efficiently. In doing so, metadata regarding which tool wafers ran on before becoming defective could also be tracked and stored to aid in root cause analysis investigations. Additionally, the GUI can easily be repurposed into an engineering tool since many engineers in the Product and Photolithography Engineering groups frequently sort through the EAGLEView server to find which logpoint defects first became visible for certain wafers. Finally, it can be used to process and display data from the new in-line inspection system summarized in Chapter 11.

4.2 Design of New User Interface

It is crucial the designed interface meets all user needs and functionalities. Additionally, it is key that this interface is developed in a robust and iterable fashion for proper implementation. In this subchapter, details regarding the design and development of the interface are covered.

4.2.1 Development Software

To develop the new GUI, Python is utilized due to each team member being relatively proficient in the language and necessary libraries like OpenCV and PIL. Additionally, the PyQT library is amongst the most commonly used libraries for GUI development and applications can easily be generated and run on other systems using a handful of commands [8].

4.2.2 Proposed Integration of User Interface

In the macro inspection module, the EAGLEView user interface does not run locally to a computer. Rather, it runs on a server that users must access individually. This leads to issues in terms of run and loading times that specialists and engineers frequently complain about. Also, whenever two or more users try to access the software simultaneously, the other user gets booted off - which is problematic. The team proposes implementing the new GUI to run locally on whichever device it is installed to alleviate this issue. Figure 4-1 below shows the current system's configuration regarding software and detection algorithms. Figure 4-2 shows the proposed system.

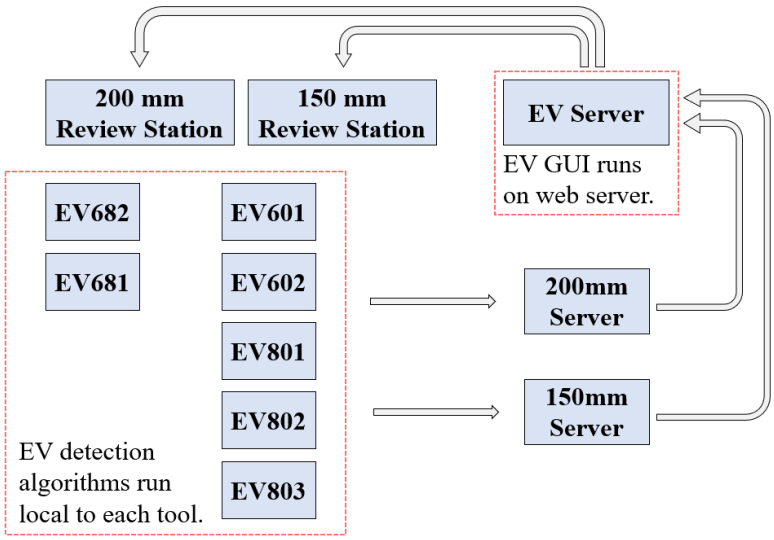


Figure 4-1: Initial macro inspection module data flow. In this orientation, EAGLEView's algorithms run locally, and compressed jpeg files classified as good or bad are uploaded to the server corresponding to the wafer size. The review stations access a web-based application that loads images from the respective server and displays information to the operators.

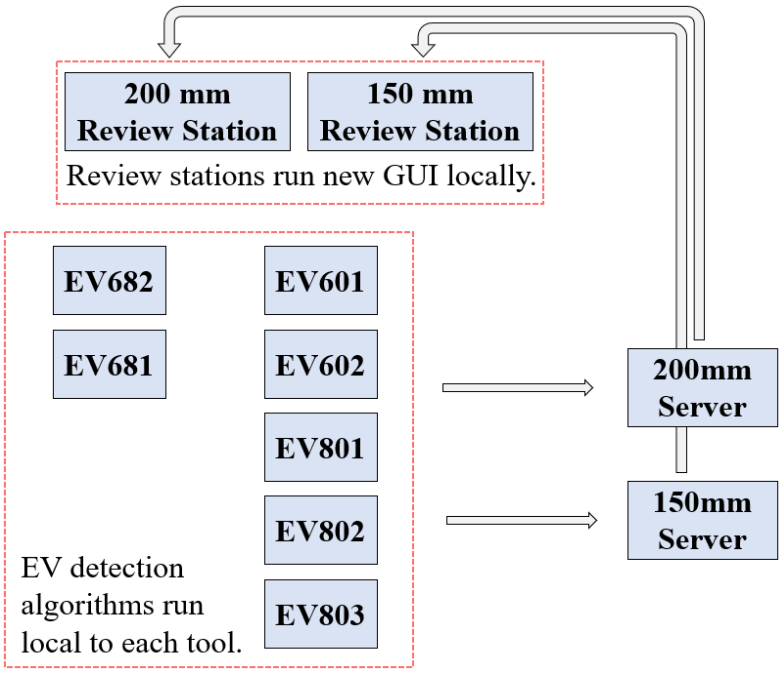
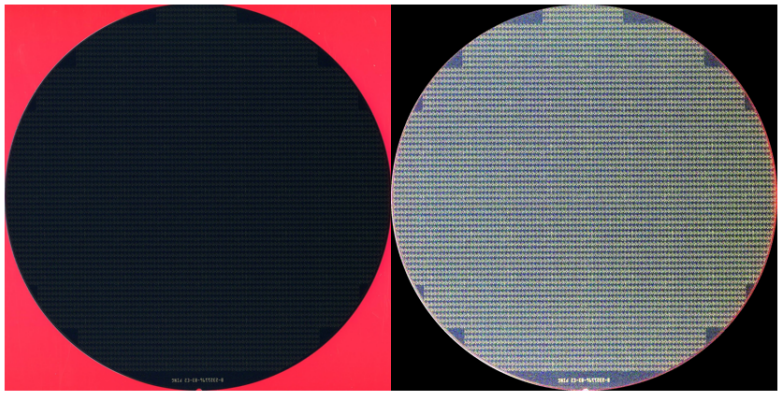


Figure 4-2: Proposed macro inspection module data flow. In the new orientation, the GUI runs locally on the review station PCs. This simplifies the data flow and fixes issues associated with multiple users and speed.

In each of the above orientations, EAGLEView runs its own detections local to each tool, and then uploads compressed versions of the high-resolution bitmaps in the form of jpegs that are approximately 2000 by 2000 pixels. From there, the existing interface software simply loads the image from the appropriate server and displays it to the specialist, where he or she decides to flag wafers for micro inspection. It is also noteworthy that the current configuration has no tie to automation. Currently, specialists have two windows open at the inspection station - one for review and one for flagging lots for micro inspection. This is problematic as specialists needlessly flag wafers inside the EAGLEView user interface, and that information is not going anywhere. Instead, hand-typed comments are sent to micro-inspect along with the lot containing instructions. User-typed notes are supposed to be consistent between specialists, but each specialist uses inconsistent language when describing inspection procedures. Problems arise in the form of miscommunication and difficulty algorithmically sorting specialist decision data. Thus, the team proposes that the new GUI be integrated with automation so that one window may be opened without typing separate notes to send to the micro inspection station.

4.2.3 Color Equalization Filtering Development

Enhanced filtering capabilities are a vital customer need for the new GUI. Engineers within the Product Engineering team have identified several methods of filtering wafer images to see defects better. These methods are only used in root cause investigations though. The most popular and universally successful method for filtering wafers is color equalization. The images can be created by using OpenCV's equalizehist function on the r, g, and b channels of the image, and then merging the channels. This method works by taking the range of values that exist within each color channel and stretching them such that the bounds are at or approaching 0 and 255. The resultant image is such that contrast is enhanced, and the image intensities are more evenly distributed [8]. Figure 4-3 below illustrates the color equalization filter applied to a wafer image of an early logpoint.



Wafer image as scanned. Wafer image equalized.

Figure 4-3: Color equalization demonstration. On the left, features of the wafer are nearly indiscernible - making specialists either blindly trust a potentially flawed EAGLEView decision or make their best guess. On the right, the color-equalized image beautifully reveals the underlying patterns present within the darker layers. Defect visibility is also greatly improved using this methodology.

The described color normalization is integrated into the new user interface through a multithreading operation so that whenever a new lot is loaded for investigation, the filter is applied to all 25 wafer images present from the EAGLEView server. Filtered versions of the images are saved in a folder created in parallel to the original logpoints folder in the server, with the only difference being the string "__FILTERED" appearing at the end of the folder name. This folder is utilized to store other information discussed in the following subsections.

4.2.4 Retroactive Logpoint Tracking Development

As mentioned previously, each lot of wafers passes through EAGLEView after the photolithography module. So, every unique layer of the wafer is scanned just after being coated with a UV-cured photopolymer that dictates circuit geometry. Because of this, every wafer should have a scan captured of it for every single layer level available on the EAGLEView database. Each layer level corresponds to a logpoint and is stored with a four-digit identifier, a date, and a time. Figure 4-4 below shows how this file structure manifests within the 150mm and 200mm databases.

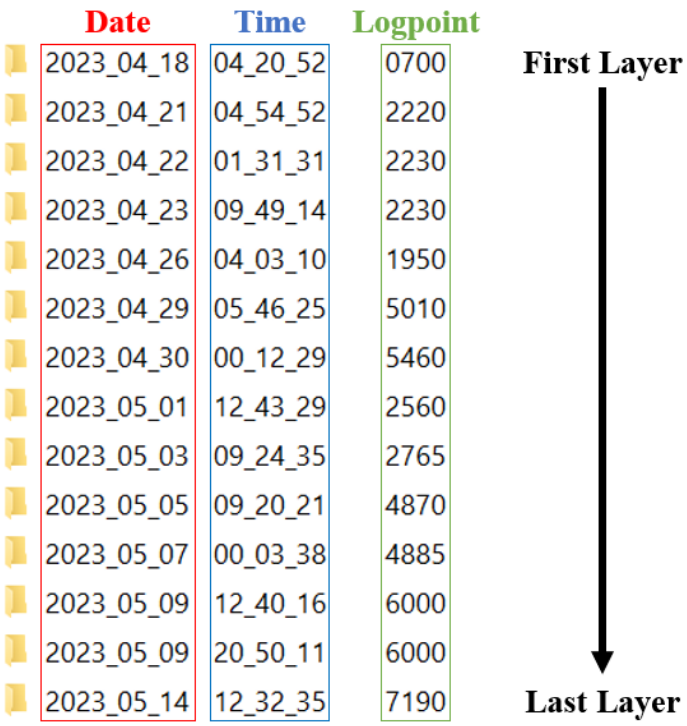


Figure 4-4: Saving structure within lot on 150mm or 200mm database. The lot information captured above is stored within a folder containing a seven-digit lot identifier number. Each logpoint is stored with a date, time, and logpoint identifier shown in red, blue, and green respectively.

Within each of the logpoint folders are scans of each of the wafers and files with information on the EAGLEView and lot. EAGLEView also randomizes the wafer order to avoid any sort of systematic variation that could arise from a given wafer always processing in the same spot relative to other wafers for a given process. Figure 4-5 below shows the naming format for wafer scans within their respective logpoint folders.

XX_[XX]_XXXX.jpg	**Examples:**
XX – Incoming Slot Number	**01_[01]_bad.jpg**
[XX] – Wafer Number	**01_[01]_bad_org.jpg**
XXXX – Good/Bad EV Decision	**01_[01]_good.jpg**

Figure 4-5: Naming format for individual wafer scans. Each filename stores the incoming slot number, wafer number, and EAGLEView classification as shown in red, blue, and green respectively. This information is extracted in the new interface to enable the display of relevant wafer information.

It is important to note that wafers deemed defective by EAGLEView have two copies. The first is called ‘bad,’ which is the wafer scan with additional marking for what the EAGLEView thinks is

defective. The second is ‘bad_org,’ which is the original wafer scan produced by the EAGLEView with no marking.

Using the naming formats, a feature for helping specialists view the same wafer at different logpoints is developed. It works by using basic string manipulation to read each wafer’s number and find the file path to that wafer present in each of the logpoint scans. Then, using the date and time within the logpoints, the file paths are sorted from recent to oldest. The user can then iterate through historic logpoints for the same wafer by using buttons or hotkeys and see how the defect and wafer have progressed over time. Additionally, the previous layer can also be filtered utilizing the color equalization filter described prior in Section 4.2.3.

4.2.5 Specialist Inspection Flagging and Tracking Development

A key requirement of the new and improved GUI is enhanced specialist tracking, flagging, and poka-yokes. Supervisors would like to know how long the specialist reviews each wafer to help determine the root cause of a defect making it through the macro inspection module. Additionally, by tracking specialist decisions and review times, metrics can be constructed to evaluate the effectiveness and efficiency of each specialist. From here, specialists with higher performance metrics can be rewarded, and their review strategies can be used to train specialists with lower scores. To achieve this, an internal timer is set within the GUI’s code to capture how long each wafer is inspected. Within each lot that is inspected, these individual times are appended to a list, and then summed and written to a file containing other inspection metrics. Similar to the filtered images, this file is saved to the “__FILTERED” folder corresponding to the lot that is inspected. This file is referred to as the “Inspect Data” file, and it is generated for every lot that a r decision is submitted on. Table 4-1 below shows an example of the Inspect Data file generated for a test lot.

Table 4-1: Inspect data file sample generated by inspection.

In Slot #	Wafer #	Out Slot #	Inspect Time (sec)	User Flagged? (val = 1)	Defect Code (0-28)
1	0	0	5	0	None
2	10	24	0.8	0	None
3	7	13	0.4	0	None
4	16	1	5.6	1	10
…	…	…	…	…	…
25	15	14	1.3	0	None

In addition to storing how long each wafer is inspected, five other columns of information are tracked and stored for each wafer. The slot when the wafer entered EV, the wafer number, and the output slot number are stored in the first three columns respectively. The fourth column displays the aforementioned inspection time associated with every wafer. The "User Flagged?" column displays whether or not the specialist flags a given wafer for a defect, where 1 corresponds to a flagged wafer. The final column, "Defect Code," shows the associated defect code a given wafer is flagged for. The team simplified the 80 plus defect codes within the current EAGLEView software to 28 unique codes as shown below in Table 4-2.

Table 4-2: Defect codes.

Defect Type	Defect EAGLEView Code	Defect Number
Scan Artifact	SCAN	0
Bubbles	BUBL	1
Ghost / Channel Stop	GHOS	2
Tilt	TILT	3
SOG	SOGP	4
Spin Defects Entire Wafer	SPEW	5
Spin Defects Center	SPCT	6
Spin Defects Edge	SPED	7
Spin Defect Single	SPSL	8
Particles	PART	9
Hotspots	HTSP	10
Flash Fields	FLFD	11
Develop Drips	DVLP	12
EBR Drips	EBRD	13
Contamination Large	CNLG	14
Contamination Small	CNSM	15
Bad Rinse	BRNS	16
Wide EBR	EBRP	17
Lifting Resist	LIFT	18
Scratches Human	SCRH	19
Scratches Machine	SCRM	20

Chips	CHIP	21
Color Variation	CLVR	22
Over Developed	OVDP	23
Partial/No Pattern	MISS	24
Lense Heating	LENS	25
Misalignment	MSAL	26
Other	-	27

The decision to simplify the number of defect codes down to twenty-eight is driven by the fact that specialists rarely use most of them. Additionally, it makes sorting of data for ML simpler by reducing the possible number of classes. Each of these unique defect types has many unique appearances and root causes. By correctly identifying defects more frequently and at a higher rate, root causes can be identified faster. As a result, systematic shifts in tool performance causing higher rates of defectivity can be resolved faster, meaning fewer wafers will be affected.

Logic is also implemented such that the submit button cannot be clicked for a lot until every single wafer is viewed by the specialist using the software. The required inspection time parameter is tunable by the manufacturing supervisor. This poka-yoke helps ensure that every wafer is reviewed by an operator, and should decrease the amount of reworkable scrap that passes through the module each month.

4.2.6 Machine Learning Recommendations Development

The most crucial element of the new GUI is real-time ML-based recommendations for specialists. To develop this feature, an ML model is first required to give inferences on what defects are present within a given wafer image. As outlined in Chapters 8, 9, and 10, several different ML models are developed by the team for different types of defects. Additionally, as discussed in Chapter 5, wafer images are deconstructed into smaller image slices. This strategy helps overcome the relatively limited amount of sorted defect data present within DFAB's databases. So, to run inferences on a wafer, the schema shown in Figure 4-6 is utilized.

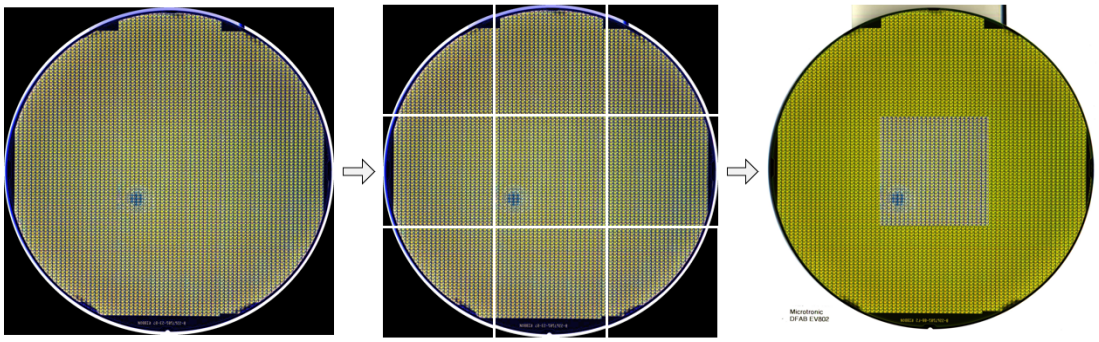


Image cropped and equalized.

Image split into N^2 slices. Inferencing run on each slice.

Defective areas identified, selectively filtered, and shown.

Figure 4-6: Image splitting, inference, and display scheme for new GUI. The color-equalized wafer is split into N^2 slices, where N is three in this case. Models run inferences on slices, and areas identified as positive (containing a defect) are overlaid onto the original image as shown on the right.

As seen above, each wafer image is broken into smaller square images. This N value corresponds to the number of squares the wafer image will be broken into along each axis. For instance, an N value of 3 results in 9 total images since the wafer is broken into three images along each axis. From here, the individual slices are run through the respective ML models. When a defect is detected, the wafer slice is put through the color equalization filter, and when a defect is not detected, no filtering is applied. The visual of the final inference wafer is then integrated to be displayed to the user in parallel to the original image.

Inferencing time is a concern when implementing this on the final production GUI. While it is advantageous for ML inferencing to run in a parallel thread within the GUI, inference time can become a rate-limiting step for wafer images split into smaller squares. This is not an issue when a small N value is utilized. As N grows, the amount of time required to inference on a wafer grows exponentially, creating a bottleneck. Since several models are to be run on every wafer, and N values above five are utilized for ML training, multithreading of inferences within the new GUI is not utilized. Instead of running the inferences within the GUI, a separate Python script is proposed that monitors the EAGLEView databases. Upon a new wafer image being saved to its corresponding folder by EV, the described splitting and inference procedure is immediately carried out, and images are saved to the "__FILTERED" folder. This is an acceptable solution given that EVs take several minutes to run a lot of wafers with each wafer image being uploaded to the corresponding server as soon as the scan is captured. Additionally, specialists do not inspect the lot until after the tool has finished running, and wafers have been unloaded from it. As a result, there is a several-minute gap between when the last scan is captured, and when the specialist opens

the lot in the user interface. Thus, inferencing can occur safely within this time frame. This feature must be developed more fully once the tool is implemented on the production floor.

4.2.7 Other Features Development

Other standard features expected for a production-grade tool are implemented into the new GUI including zooming, panning, fullscreening, and more. The use of these features is discussed in Section 4.3 of this chapter, but development is not detailed in this work.

4.3 OwlView Interface

The current version of the OwlView interface is introduced in this subchapter. Some of its key features are highlighted, such as color filtering, navigation, lot loading, and retroactive logpoint tracking.

4.3.1 OwlView Interface Layout

An annotated view of the new GUI, called OwlView, is shown below in Figure 4-7. As shown, OwlView has five main sections. The "Wafer View'" section shows the current wafer loaded into the lot by the user. The "Recent Logpoints" section shows the most recent logpoints loaded in from the EAGLEView server. The "ML Section" shows the recommendations generated by ML models discussed in Chapters 8, 9, and 10. The "Wafer Navigation" section shows information the specialist can utilize to navigate through and inspect wafers. The "Defect Flagging and Lot Submission" section gives an interface for the user to flag defective wafers. The use of each of these components is discussed in the following subsections.

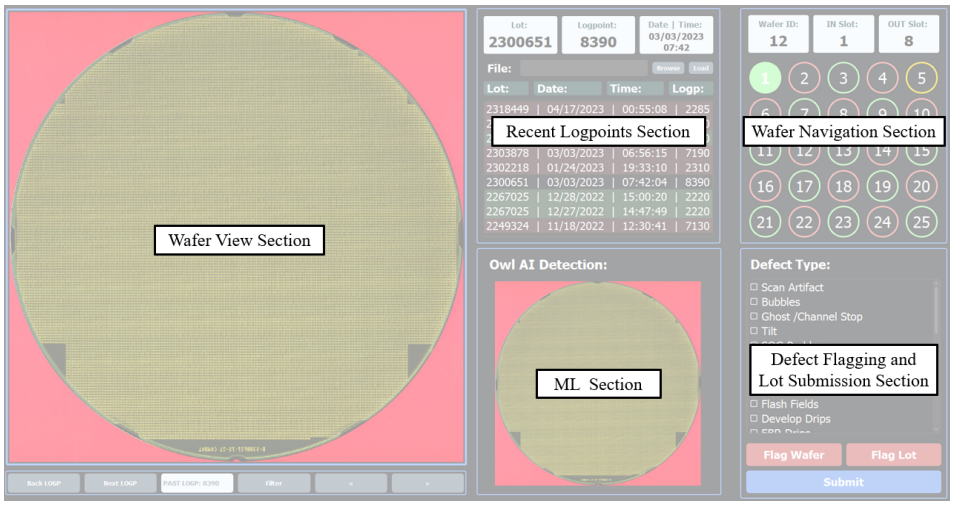


Figure 4-7: OwlView interface with annotated sections: OwlView's five main sections are outlined.

4.3.2 OwlView Wafer View Section

The wafer view section of OwlView is the largest component within the GUI window. Here, specialists can view each wafer image in a large screen format. An expanded view of this section is shown below in Figure 4-8.

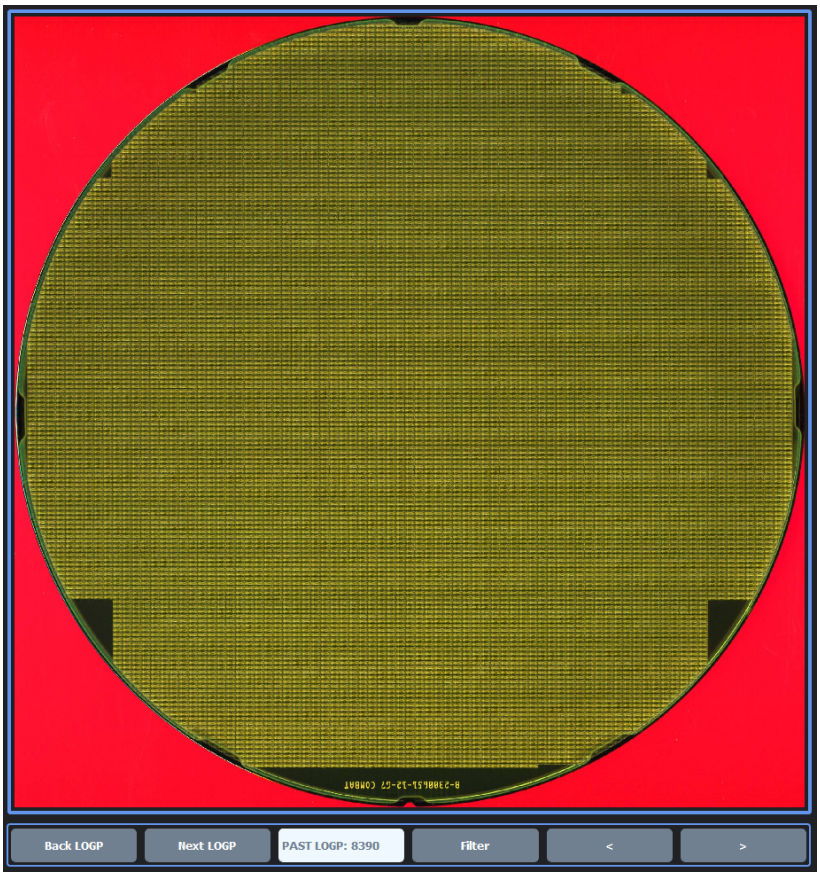


Figure 4-8: OwlView wafer view section: A focused view of the wafer view section of OwlView is shown above. In this section, the user can zoom into wafer images, apply filtering, move between wafers in a lot, and move between logpoints for the same wafer.

To navigate between wafers, the user can click on either of the left or right arrows at the bottom of the screen. To apply the color equalization filter, the user can select the filter button at the bottom of the screen. To see the wafer at previous logpoints the user can select the "Next LOGP" and "Prev LOGP" buttons. Hotkeys are also available such that the operator may complete all of these functions with intuitive keyboard commands. This is crucial given the large volume of wafers a specialist reviews within a given day. Figure 4-9a through Figure 4-9c below shows the functionalities of this section.

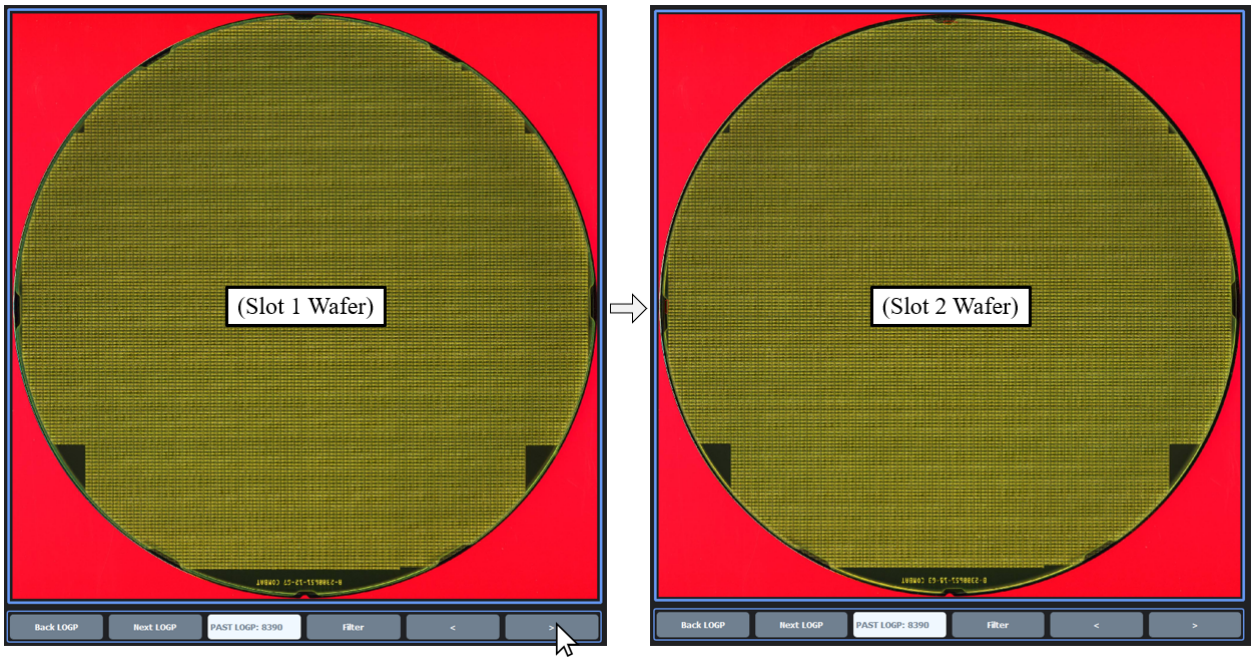


Figure 4-9a: OwlView wafer view section functionality: wafer navigation. To move between wafer slots, the user can click one of the two arrows shown in the bottom right of the window.

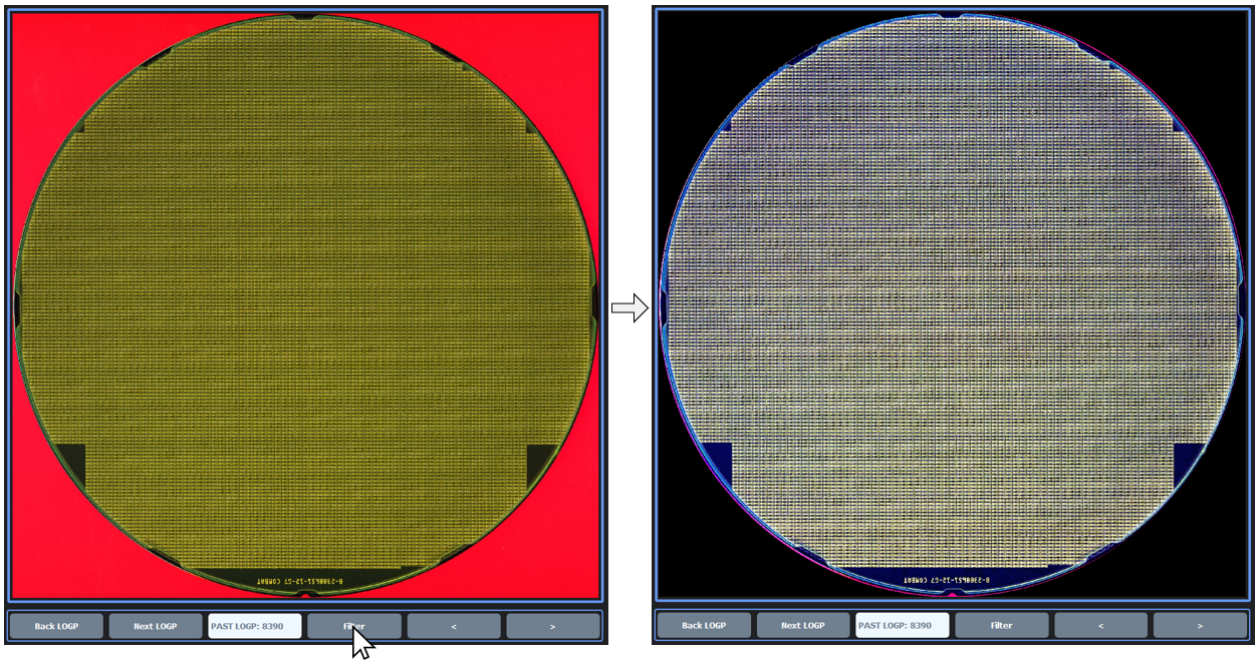


Figure 4-9b: OwlView wafer view section functionality: wafer filtering. To filter wafers, the user can click the button labeled “Filter” in the bottom of the window.

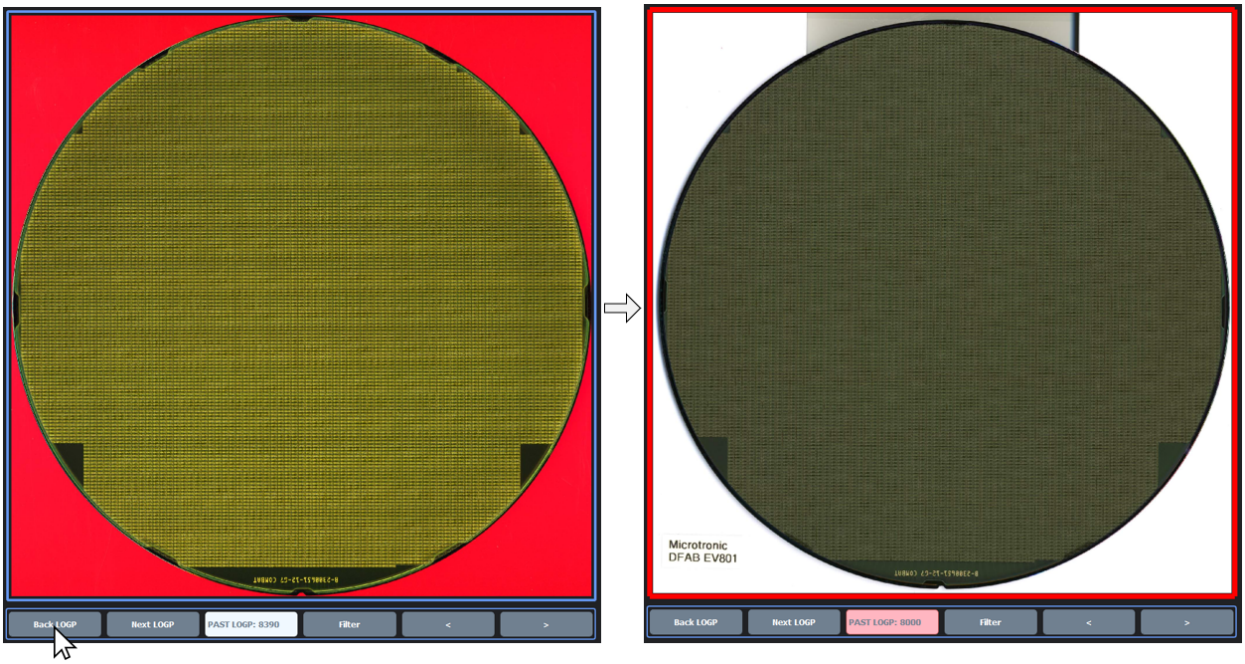


Figure 4-9c: OwlView wafer view section functionality: logpoint tracking. To change logpoints, the user can click the button labeled “Back LOGP” to go backward, and “Next LOGP” to go forward.

Additionally, the user can zoom into the wafer image by double-clicking with their mouse. Panning is also available. The integration of these features alone is a significant improvement to the current user interface.

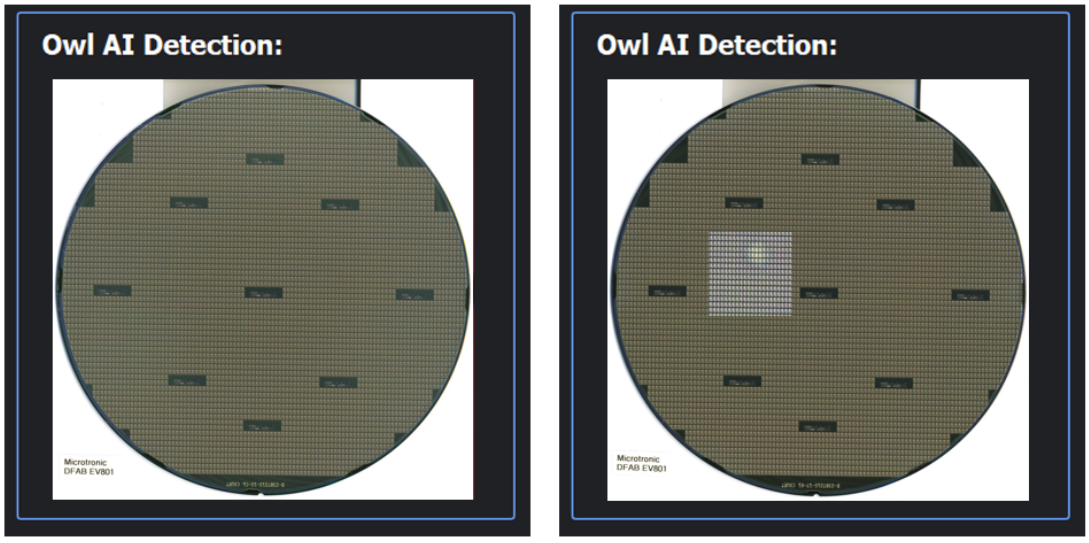
4.3.3 OwlView Recent Logpoints Section

In the recent logpoints section, the user can see what logpoints have been scanned most recently by the EV. An expanded view of this section is shown below in Figure 4-10. Here, the most recently scanned logpoints continually filter to the top of the queue. The specialist can load in the lot for a given logpoint by clicking on the desired row and hitting the load button. Alternatively, the user may load in specific logpoints by clicking the browse button and looking at the server's file system if desired. Further, the user can type in the seven-digit lot identifier number to the top text bar if desired. Once a lot is inspected and submitted by the operator, it will appear green if no wafers were flagged for inspection, and red if any wafers failed. This functionality ensures that operators can efficiently and effectively load in and keep track of inspected lots. For user convenience, the lot’s information is displayed in large characters at the top of the screen, whereas it is obscure to find in the current software.

Figure 4-10: OwlView recent logpoints section. In this section, the user can load in recent logpoints.

4.3.4 OwlView ML Recommendations Section

In the ML recommendations section, the user can see a specially filtered image generated based on ML inferences. An expanded view of this section with examples of flagged and not flagged wafers is shown below in Figure 4-11.



Good wafer not flagged. Bad wafer flagged.

Figure 4-11: OwlView ML section. In this section, the user can see selectively filtered images generated by ML algorithms. The above example is generated based on training data.

As the user moves through the wafers of the lot, the ML-filtered image is automatically shown in parallel with confidences for each of the classifications. This filtering method, as discussed in Section 4.2.6, varies from the EV-generated images, as rather than outlining defective pixels in red, they are simply color equalized. This is superior because EV-generated images often interfere with the user's ability to truly tell if there is a defect present. Given EV's poor performance on GR&R, this leads to an abundance of falsely flagged good wafers where the specialist may simply blindly trust EAGLEView and send the lot to micro inspect. When the ML algorithms falsely flag a wafer, the underlying area is more visible than before, making it such that the specialist can make an informed decision to pass or let the defect through.

4.3.5 OwlView Wafer Navigation Section

In the wafer navigation section, the user can see and move between wafers based on the incoming slot numbers. An expanded view of this section is shown below in Figure 4-12. At the top of this section, the wafer number, incoming slot number, and outgoing slot number are also plainly displayed and updated continually for every wafer. This helps the specialist identify which wafer to take out of the cassette when necessary. The circular numbered buttons correspond to each slot of the cassette. These buttons are outlined blue if the slot is empty, green if EAGLEView classified the wafer as good, red if the EAGLEView classified the wafer as bad, and gold if the wafer is the golden wafer. This intuitive display helps the specialist understand why EAGLEView made the classification it did. For instance, if every wafer is marked as red but the golden wafer, there is likely a defect on the golden wafer. If every other wafer slot is marked as red, and EAGLEView is right, there is likely some systematic variation in tool performance affecting every other wafer. As this information is tagged by the operator it is saved in the Inspect Data file and can be utilized for future ML training.

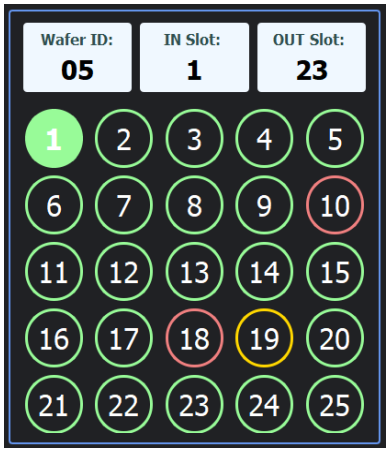


Figure 4-12: OwlView wafer navigation section. In this section, the user can see relevant information and navigate between wafers graphically. Green circles are good according to EV. Red circles are bad. Gold is the golden wafer.

Whenever the wafer is currently being inspected and displayed in the wafer view section of the GUI, the corresponding button has a solid fill. Once a wafer is viewed, the fill of the button becomes transparent but visible. This way, the specialist can keep track of what they have and have not reviewed. Figure 4-13 below shows this functionality, where every wafer after wafer 12, which is currently being viewed, has not been inspected.

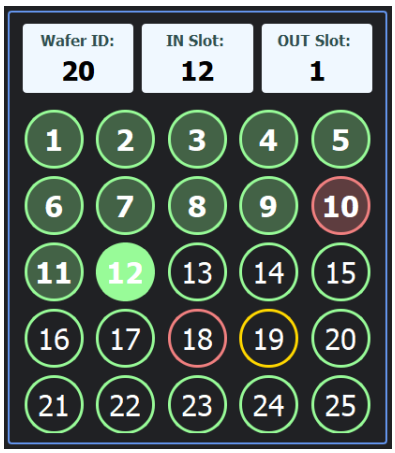


Figure 4-13: OwlView wafer navigation functionality. Once a wafer has been viewed, it gets a semi-opaque background. The user must review every wafer to submit the lot.

In the configuration above, the user cannot submit the lot until every wafer has been inspected and has a partially opaque infill.

4.3.6 OwlView Defect Flagging Section

In the defect flagging and lot submission section, the user can flag wafers for inspection and submit the lot. An expanded view of this section is shown below in Figure 4-14. Here, the 28 unique defect codes are present for the specialist to select and apply to each wafer. To apply a defect code to the wafer, the user must simply select the check box corresponding to the defect and select the flag wafer button. When the user does this, the wafer image in the wafer navigation section will turn a bright pink color. This is exemplified in Figure 4-15.

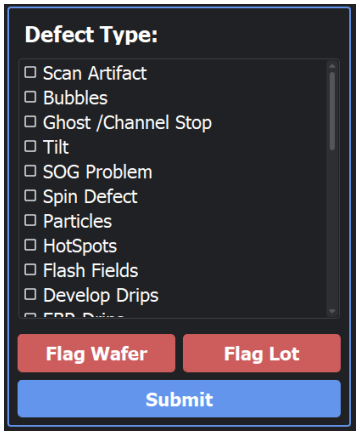


Figure 4-14: OwlView defect flagging section. Using this section, the user can select different defect classes and apply them to different wafers in the lot.

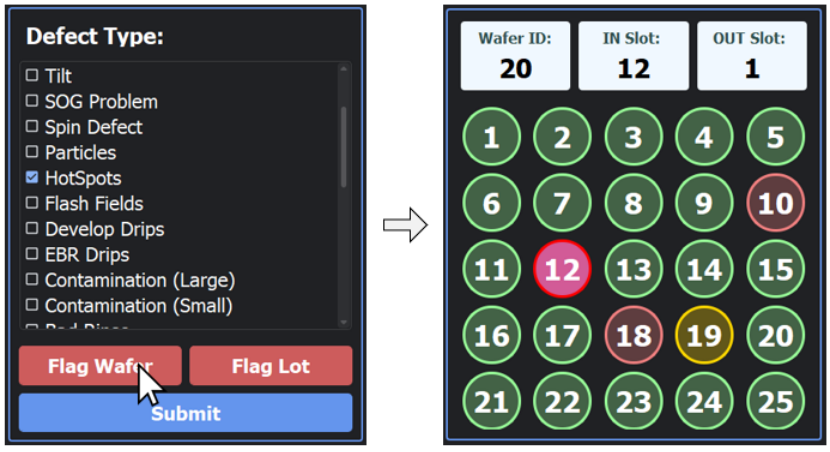


Figure 4-15: OwlView defect flagging functionality. In the above example, the user flags the wafer in slot number 12 as having a hotspot defect. They do so by clicking the “HotSpots” checkbox and the flag wafer button next. The wafer navigation section updates by highlighting the button bright pink.

If the user wishes to unflag a wafer, he or she can simply view the wafer in question and reclick the flag wafer button. This will undo the changes brought about by initially clicking the button. If the user wishes to flag a whole lot, the same process holds, but with clicking the flag lot button. Also, whenever the specialist moves between wafer images within a lot, the checked defects update. Whenever the specialist is finished inspecting a lot, he or she can select the submit button. Then, the Inspect Data file is created and populated with the appropriate defect codes. In addition to the Inspect Data file, a metadata file is generated containing information on which tool the lot was run on before EV, what technology group the wafer is a part of, and more. This can be utilized by engineers for root cause analysis, and be fed into future ML models for enhanced root cause finding. Finally, whenever users open a lot that has already been inspected, all of the previously flagged wafers will be loaded in and can be modified.

4.4 Implementation of OwlView

Implementation of the OwlView interface as a production, engineering, and ML tagging tool are discussed in this subchapter. OwlView provides benefits to each of these three avenues that are described, along with actions taken by the team to achieve implementation.

4.4.1 Use as Production Tool

Steps have been taken to get the new OwlView interface into production. This is important as the enhanced features will help with specialist accuracy. Additionally, as specialists utilize the tool in production, the availability of tagged data for machine learning skyrockets and can be fed back into the ML models.

The SOPs for the current system were read and then reworked by our team for the new OwlView interface. In the new procedures, tool use is described per standard TI formatting. Additionally, errors in the old SOP are corrected. For instance, in the old SOP, it was discovered that specialists let through defects if they affect less than one percent of the wafer. This heuristic allows SOG defects to be let through production by specialists even if they do see them. In the new SOP, this feature does not exist, so the SOG defect catch rate should improve.

In addition to new SOPs, training documents were created by our team for the specialists and supervisors. Several training sessions were conducted with specialists across shifts in which we gave specialists hands-on training in the software. Feedback from specialists and supervisors was generally positive, and new features were added as a result of our discussions.

The OwlView Python script was also compiled as an application file that can be run on any PC with Windows 10+. Unfortunately, the PCs on the factory floor are not up to date. As of 08/11/2023, the

photolithography and metrology teams are working on a PC upgrade so that this software can be utilized on the production floor.

4.4.2 Use as an Engineering Tool

The OwlView interface is also helpful for engineering teams. Engineering teams across the photolithography and product groups frequently do root cause investigations where they inspect hundreds of lots that passed through EV. In doing so, they manually sort through the EAGLEView servers, copy files over, and run filtering scripts to look for defects. To help aid this process, a version of OwlView is given to engineering teams for enhanced root cause identification.

4.4.3 Use as a Machine Learning Data Tagging Tool

OV is also a powerful ML data tagging tool. Local copies of lots with known defects are first copied to the user's computer and then accessed using the browse feature within OV. Next, the user can inspect lots just as a specialist would. From here, information from the inspect data file is used to sort data into classes based on the tagged defects.

This is important to highlight since the data infrastructure within DFAB makes it challenging to find large quantities of tagged data. From the moment this GUI is implemented on the floor for production, Inspect Data files will be generated by specialists nonstop, greatly multiplying the potential for engineers to train more robust ML models with a broader array of representation for all defect types across more technologies.

4.5 Chapter Summary

To give real-time ML suggestions and improve specialist performance, a new GUI called OwlView was developed by our team. OwlView contains enhanced color filtering, logpoint tracking, ML recommendations, and greater functionality overall compared to the current software system. SOPs and training documents were generated for the software package. Changes to the IT infrastructure were also implemented so that the software can be integrated in-line with production. Additionally, training sessions were conducted by our team with specialists. Along with the enhanced functionality provided by the use of this software in production, it has powerful applications as an engineering and ML data tagging tool.

Chapter 5

Data Availability and Data Preparation Tools

This chapter details data availability and tools developed to tag and sort data for machine learning models. The motivation for a robust ML infrastructure is first described, followed by an overview of data availability at DFAB. Scripts for tagging, sorting, and splitting the data are also covered.

5.1 Motivation for ML Data Infrastructure

Before our work, DFAB had no infrastructure for data collection or tagging. Procedures and tools for gathering and tagging large amounts of image data are crucial to training models. Further, a robust ML infrastructure applies to any image, such as those collected in the in-line inspection trial as discussed in Chapter 11 of this work.

5.2 Data Availability

Wafer scans captured at the EAGLEView module are stored within one of two databases dependent on wafer size. Within each of these databases, there are two subdirectories: one is a current database and one is an archive. The current database stores wafer images from the last six months approximately, whereas the archive stores them for about a year. Eventually, wafer images are lost to deletion to save server space. The general flow is from the current database, to the archive, to deletion as shown in Figure 5-1. Within this structure, data is sorted by lot and logpoint with EV's good or bad classification. Due to the poor performance of EV, there is no meaningful way to automatically extract and sort data for ML training.

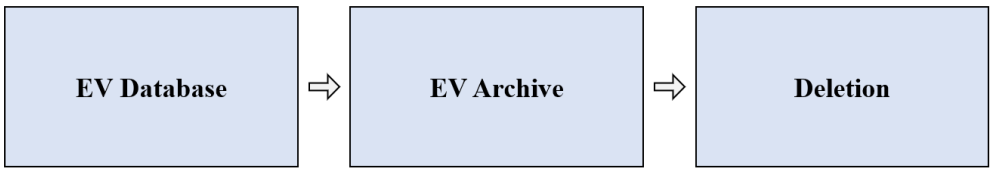


Figure 5-1: Wafer scan location through lifetime. EAGLEView scans are moved from the active database to an archive after being inactive for several weeks. From there, the archive is cleared as more storage space becomes necessary.

It was suspected that SQL queries could be generated to pull lot and logpoint numbers from the micro inspect station that was reworked. Unfortunately, this is not feasible. Although the micro inspect station decisions are stored, the wafer numbers affected and defect type are not. Additionally, all notes regarding this are hand typed by the specialist, with each one using slightly different words and codes to describe the same issue.

Several good sources of confirmed defect data are present, however. The first is engineering containment data. Whenever a certain tool error is causing systemic defect propagation across many wafers, engineers open contaminants in which they manually inspect dozens and even hundreds of wafer images for suspected defects. They track these defects, the wafers affected, and lot information so that they may be identified in the EAGLEView server. Information regarding the lot, wafer numbers, and defect types are stored in a file.

A second source of data is the photolithography rework report - which is used by engineering management to evaluate tool performance. While powerful, data tagging is not standardized, and many lot numbers described within these reports are not actually affected by the described defect classes. Additionally, of the 80-plus defect types, only a handful have meaningful amounts of data.

A third source of data is online reports. These allow users to search for event types that have led to a large amount of wafer scrap. While useful in some cases, only events that have occurred within the last year are useful since older EAGLEView scans are deleted.

From each of these data sources, files containing lot and logpoint numbers for lots affected by certain defects can be populated. Data does have varying sparsity for different defect classes, however, and the wide array of sources makes it difficult to gather large amounts of affected wafers. Additionally, due to the error rate and lack of wafer number-affected labeling, manual tagging is required to ensure proper data sorting.

5.3 Scraping from EAGLEView Server

Once a file containing lot and logpoint data of lots containing defects is gathered from one of the sources described in Section 5.2, those lots are copied from the EAGLEView server to a local destination on the user's computer. To do so, a basic Python script is constructed that searches through both EAGLEView servers and writes local copies of the images to the user's computer with the same organization present on EV. Next, these images must be tagged for defects.

5.4 Data Tagging

In this subchapter, procedures and tools developed for data tagging are outlined. These include the procedure for using OwlView to tag data, as well as the separate spatial defect tagger developed by the team.

5.4.1 General Defect Tagging

With local image data saved to the engineer's computer, he or she then tags data using the OwlView interface described in Chapter 4 of this thesis. To do so, the engineer loads in a lot of wafers and inspects them just as a specialist would. After submitting each lot, an Inspect Data file is created containing all of the engineer's decisions. To ensure accurate tagging, team members studied the same SOPs and training material specialists use to identify defects. Additionally, because these lots have defects confirmed for one reason or another, the likelihood of wrongly classifying a defect is reduced. If at any time a team member is unsure of a defect, he or she can check the "maybe" button within the defect classes, making it so that the data sorting software bypasses the wafer in question.

5.4.2 Spatial Defect Tagging

For each wafer confirmed as having a defect, a separate spatial defect tagging tool is developed and implemented by our team. This interface works by opening all images in the defective wafers folder, as sorted in Section 5.4.1, and allowing the user to draw a mask showing where the defect is located. This interface and its use are demonstrated in Figure 5-2 below.

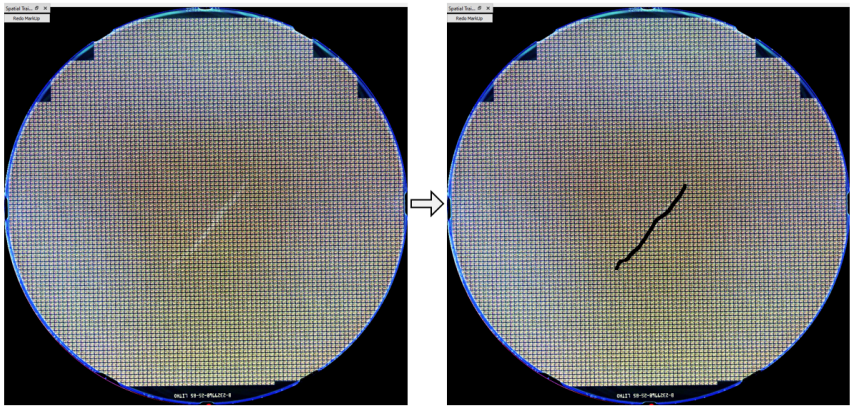
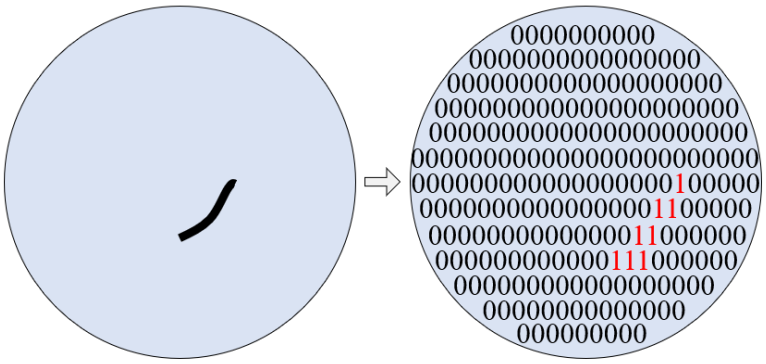


Figure 5-2: Spatial trainer demonstration. In the left image, a defect resembling a scratch is present on the center of the wafer. The user marks the defect by drawing over it with their cursor (right).

In the instance above, the wafer is tagged as having a scratch defect, as seen in the left center of the wafer. The right image shows a mask generated by the user to identify where the defect type is present. The user can create a mask for any number of defects that occur in a given image. The masks are stored in files within the same folder for each image. The files have an entry corresponding to every pixel within the wafer image, where unmasked areas equal zero. Each separate mask drawn by the user is correlated to a new number, with the first mask being one. Figure 5-3 illustrates this concept.



Defect marked by user. Mask generated for training.

Figure 5-3: Mask generation in spatial tagger. Everywhere the user draws on the spatial tagger creates a mask with a non-zero value. In the mask, each pixel corresponds to a value.

Spatial tagging is important because masks enable defect localization for certain ML models. As discussed in Section 5.5.1, this method is also utilized to overcome the sparsity of defect data available to our team.

5.5 Data Sorting Tools

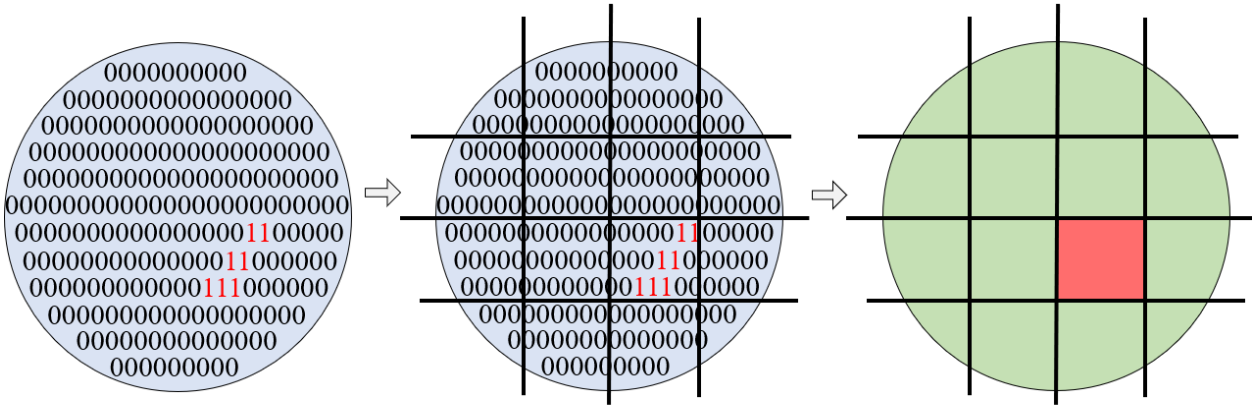
In this subchapter, data sorting tools are covered that enable the team to train effective machine learning algorithms. These include scripts for splitting the data into smaller units and resorting, and other tools for creating a balanced data set for machine learning.

5.5.1 Data Splitter

There is no reasonable way to source enough data to train robust ML models with the methods described to this point. This is especially clear after tagging and sorting data for SOG defects, which are notorious for being amongst the most difficult to spot for their small size and mild coloring. After filtering through several engineering contaminants that involved the tagging of nearly 200 lots, there are only 300

wafers identified as having defects. Additionally, these defects occur almost exclusively on the same mask level and are often obscure due to their deceptive appearance. The team observed that training a model on this data set would lead the model to identifying other patterns (such as the layer level affected being biased towards containing SOG defects). Further, there are concerns regarding the loss of features in reducing the image's original sizes of approximately 2000 by 2000 pixels. This is discussed in more detail in Chapter 8.

To overcome these concerns, a method of data multiplication is devised through our spatial tagging tool. We hypothesize that by splitting a single wafer image into smaller squares and then resorting to the squares where defects are present, we can multiply our training data set and overcome issues surrounding image compression. This idea is shown in Figure 5-4 below.



Wafer image and mask. Split based on N=4. Becomes 15 good, 1 bad.

Figure 5-4: Splitting algorithm methodology: The wafer is split based on the user-generated mask and the inputted N value. Higher N values would lead to more bad samples being generated, as the above defect would be split into more squares.

To achieve this functionality, we use a new Python script for sorting that takes a user input of N, where N corresponds to the number of squares desired along the x and y dimensions. For instance, the figure above shows how a split occurs for N = 4. Here, a single bad data point is turned into sixteen data points, where one is bad. This strategy is also advantageous because defective wafers often contain many defects spread across the wafer surfacer. Therefore, one wafer image can yield many good and bad sections. By splitting images, we work against the development of overfit biases in ML models that may result from such cases. This innovation is effective in helping models converge, as discussed in the ML sections of this paper.

5.5.2 Indices Remover

Because certain defect classes tend to affect certain areas of the wafer, tools are developed for removing unwanted squares created when using the splitting script. Whenever a wafer is split and resorted, a row and column identifier number is added to its filename. A Python script that removes user-specified unwanted indices from the data set, whether they be central or on the edge, is developed and utilized for this.

5.5.3 Random Reducer

Data sets are always imbalanced after tagging with far more goods than bads. To overcome this, a Python script for randomly reducing a data set to match the other one is produced. This is essential as training on a data set containing an imbalance of classes may guide a model into simply guessing which class is more statistically probable rather than considering specific features.

5.5.4 Train Test Divider

To train an ML model with a given data set, data is divided into testing and training folders. A simple Python script that randomly resorts data into this structure is generated. To run the script, the user must simply input the desired test and training percentages.

5.6 Chapter Summary

Within this chapter, the availability of data was first described. From there, methods surrounding tagging data were outlined. Next, the key innovation of splitting wafer images to overcome data scarcity was described. Lastly, other tools for data sorting and quality were presented.

Chapter 6

Related Work in Macroscale Defect Detection

This chapter details related work in macroscale defect identification and classification in industry and research settings. Macroscale defect detection is typically achieved using rules-based approaches in industry. Much work is being done by academics to classify defect types based on wafer yield maps using ML.

6.1 Macroscale Defect Detection in Industry

Macroscale defect detection is critical in the semiconductor industry, as reworkable defects can be caught and addressed. Before automated visual inspection, all inspection of wafers was conducted manually, but over the past decades tools have been adopted that rely on various automated inspection techniques [9]. These inspection tools can run in or out of line with toolsets, with inline inspection being superior due to a reduction in tool changes and cycle times experienced by each lot of wafers [10]. Regardless, innovations in automated visual inspection of wafers fabs helped alleviate the rate-limiting and subjective steps involving human judgments. A wide array of tools exist for macroscale defect detection that are deployed across fabs globally. At DFAB, EAGLEView is the only toolset used for macroscale wafer inspection, and it works in tandem with human operators.

Algorithms utilized by tools are typically rules-based. In the EAGLEView case, the tool has preset thresholding parameters and tolerances that dictate how much each wafer scan may vary from the initial wafer, also called the golden wafer. The primary parameter utilized for comparison is the intensity of each pixel [11]. If there is too large a variance between the intensities of corresponding pixels of wafer scans, those areas are identified as defective by the software. Other rules-based detection algorithms work without using a randomly selected golden wafer for a lot. In these cases, single images are broken into sections, and average pixel parameters are compared for each region. If a region has abnormalities compared to regions within the same region, it is likely a defect is present [12]. Rules-based approaches are powerful but not without flaws. Image resolution must be high, image quality and camera positioning must be consistent, and practical rules must be user-set that work for often unseen and unpredictable data.

6.2 Machine Learning Macroscale Defect Detection

The use of ML models to identify macroscale defects shows promise in overcoming the limitations of rules-based approaches. Several studies have been conducted for identifying and classifying macroscale defect types using the WM-811K wafer data set. This data set contains 46,393 lots, leading to 811,457 unique wafer maps for training [13]. Wafer maps show representations of die affected by a macroscale defect, and differ from actual scans of a wafer. Figure 6-1 below shows wafer maps from the WM-811K wafer data set compared to data from EAGLEView utilized by our team for training.

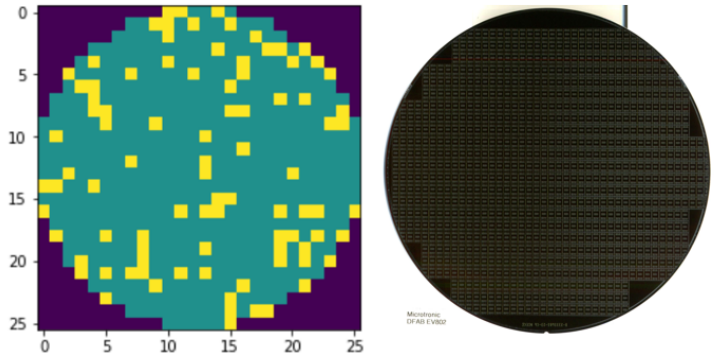


Figure 6-1: Wafermap compared to EAGLEView scan. Wafer maps are representations of the spatial distribution of defective die found on wafers. Wafermaps can be generated from wafers once they are probed at the end of the line.

Gong & Lin trained supervised learning models on the WM-811k data set using AlexNet and VGG16 architectures that achieved 88.5% and 92% accuracies respectively [14]. Additionally, Chen et al. achieved 97% accuracy in defect classification using a multi-feature fusion perceptual network [15]. These models simply classify the defect pattern present on a given wafer map, but localization is also a key feature in root cause investigation. To this end, Shinde et al. achieved 94% classification accuracy using a You Only Look Once (YOLO) architecture on the same wafer maps data set [16].

While methods utilizing CNNs have surpassed traditional methods in accuracy and classification time, they are susceptible to the effects of training with imbalanced classes - which Chen et al. addressed with a CNN-based knowledge distillation technique [17]. Boning et al. address the class imbalance issue through synthetic sample generation and integrate a selective learning approach to refrain from classifying high-risk defect patterns to achieve 99% classification accuracy of defects [18].

Other approaches employ unsupervised learning methods to identify defect patterns on unlabeled data sets. Liukkonen and Hitunen used k-means clustering and self-organizing maps to find systematic

defect patterns from large groups of wafers [19]. This methodology varies from the aforementioned supervised learning techniques that require large amounts of tagged data.

The previously described models are all trained on the WM-811k data set, whereas our work is related to real wafer images. We aim to apply techniques and methodologies described in these papers to real wafer images at DFAB.

6.3 Chapter Summary

In this chapter, macroscale defect detection in the semiconductor industry was introduced. The advantages and drawbacks of using ML over a rules-based system were considered. Lastly, attempts by others to classify macroscale defects were covered.

Chapter 7

Overview of Machine Learning Methods

This chapter overviews the machine learning architecture, hyperparameters, and strategies deployed for different defect models. Models are developed using PyTorch [20].

7.1 Selected Model Architecture and Functions

In this subchapter, the selected model architecture and accompanying functions are described and justified. Details surrounding our supervised transfer learning approach, as well as the optimizer and loss function selected are outlined.

7.1.1 Supervised Deep Learning Approach

There are two types of deep learning: supervised and unsupervised. The learning in this work is supervised, in that data with classifier labels is fed to the models. Unsupervised techniques, such as k-means clustering, require no tagging of data. In a supervised learning scheme, the objective is for a predictive model, f, to correlate data (X) and its corresponding tags (Y). The following equation displays this idea, where Θ are the parameters of the model.

$$\widehat{y}_i = f(x_i; \Theta) \tag{7.1}$$

Once the proper model is constructed, it can then be utilized to make predictions on unseen data X. From there, the effectiveness of said model is judged based on metrics discussed in Section 7.3.

7.1.2 Model Architecture

A VGG (Visual Geometry Group) deep convolutional neural network architecture is selected and utilized for models trained in Chapters 8, 9, and 10 of this work. It is utilized for its relative simplicity, wide availability of pre-trained variants, and computational simplicity when compared to other models. While true, the architecture is powerful and widely used since Simonyan and Zisserman proved the architecture's improved performance due to its deeper weighted layers paired with 3x3 convolution filters [21]. Figure 7-1 below shows a visualization of this architecture.

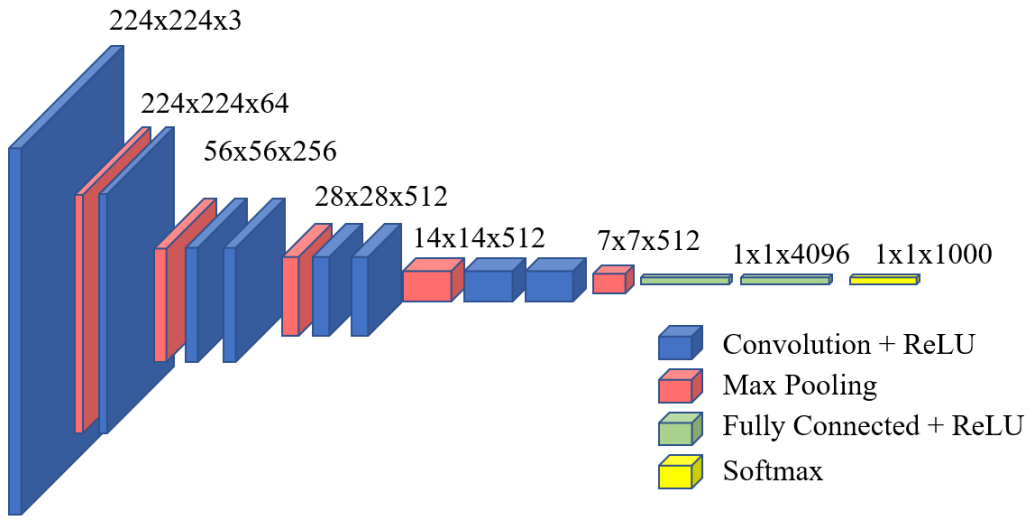


Figure 7-1: VGG-11 architecture visualization [21]. The above diagram shows the layer-by-layer breakdown of VGG-11 from the input image size, convolutions, max pooling steps, fully connected linear layers, and the softmax function.

For the scope of this work, the VGG-11 variant of the architecture is utilized, which contains eleven weighted layers, of which eight are convolutional and three are fully connected. While smaller than the VGG-16 and VGG-19 versions, it is still large and robust, carrying with it 133 million parameters [21]. The architecture is implemented through the use of easily accessible and documented libraries within PyTorch [22]. As seen in the above figure, the ReLU (Rectified Linear Unit) activation function is utilized after each convolutional layer to enable non-linear relationships to exist within data points. Pooling layers are also employed that reshape and downsample the inputted 224 by 224 by 3 images as features are extracted. In the final three fully connected layers, the linear operations are carried out and results are fed into the softmax function. Here, class scores are converted into probabilities which are then turned into binary classifications through the argmax function. The softmax function works by finding $\sigma(z_i)$ for each element in a given input array through the following equation.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{7.2}$$

Here, the input array is the output of the fully connected dense layers, of which there are two for a binary classification model. By normalizing the matrix values applied to the exponential function, confidences are constructed for both the positive and negative classifications that sum to one.

7.1.3 Transfer Learning

Due to the relatively limited availability of defect data, transfer learning is utilized to create a more robust model. When transfer learning is applied, the weights and biases of the model are initialized to those of the model trained for object recognition on a completely different data set. Yosinski et al. showed that utilizing transfer learning increases model performance when compared to using randomized weights and biases initially [23]. For our models, pre-trained weights and biases are utilized for a model trained on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017 data set. This data set contains approximately 1.28 million training images, 50,000 validation images, and 100,000 test images [24]. All models trained utilize pre-trained weights and biases.

7.1.4 Optimization

The Adam (Adaptive Moment Estimation) optimization algorithm is utilized across models developed in this work. While other optimization algorithms are applicable, Adam is selected for its automatic learning rate adaptation and momentum optimization. It works by continually updating parameters like time step, gradients, biased moment estimates, and learning rates based on previous time steps after each iteration the parameter vector, theta t, has not converged. Additionally, Adam only utilizes first-order gradients which require relatively little memory [25]. When comparing the performance of different optimizers, it has been shown that the optimal choice is often data set dependent [26]. While true, Adam is the only optimizer explored in the scope of our work.

7.1.5 Loss Function

The cross-entropy loss function is utilized for the models developed through the scope of our work. It serves as the objective function for the model to minimize as weights and biases are tuned between epochs. It works by taking the output of the softmax function, which in the binary classification case of our models, are one-dimensional matrices with two values. The cross-entropy loss function for a general classification case is characterized by the following equation.

$$L_{CEi} = -\sum_{i=0}^{K} logP(y = y_i \mid x_i, \Theta) \tag{7.3}$$

7.2 Investigated Model Parameters and Techniques

Between models trained, different model parameters and techniques are utilized in the creation of a given model. In this section, the parameters and techniques as investigated and tested by the team are outlined.

7.2.1 Regularization

Several regularization methods are deployed to prevent overfitting. The first of these is dropout, which is deployed in the fully connected layers of our models. Dropout functions by stochastically dropping units and corresponding connections from the model during training, and has been proven to reduce the effects of overfitting in vision applications [27]. All models trained in this work utilize a dropout value of 0.5.

A second method is the integration of early model termination. Under this scheme, save states are continually generated for a model during training between epochs. These save states are based on metrics such as loss, validation accuracy, f1-score, and so on. By implementing this technique, models can be saved before entering overfit territory.

The third method utilized is weight decay. Weight decay penalizes model complexity and thus, overfitting. When weights are updated by the optimizer, an additional term is added in which two times the product of the initial weight and weight decay coefficient is subtracted from the weights. Several weight decay coefficients are explored in this work to protect against overfitting.

7.2.2 Cross-Validation Technique

While some SOG models discussed in Chapter 8 of this work use a typical train and test data split for training, all other models utilize a cross-validation technique. Cross-validation differs in that the training data set is split into different folds, where the amount of training data in each fold is the total data in the training set divided by the number of folds. When training, a different one of the folds is withheld from the training set each iteration. The withheld data set then becomes the validation set that the model inferences on each epoch for feedback into the loss function. The test set initially withheld is used to inference across all folds at the end of training. This method helps more robust models be designed and selected [28].

7.2.3 Image Split Value N

As introduced in Chapter 5, image splitting is utilized to overcome the limited nature of tagged and sorted defect data. In this scheme, the defective wafer image is divided into N^2 subsquares, where N is the number of squares along each image axis. As N is increased, the total number of images grows to the square power, increasing the effective data set and reducing the image's resolution. This is advantageous as the VGG-11 architecture takes a 244 by 244 pixel image as an input, so a smaller split image maintains less feature loss in the resizing process. The N value is modified depending on defect size and availability. SOG defects, discussed in Chapter 8, are amongst the smallest of defects, and a larger N value is required as a result. Edge and center defects, investigated in Chapters 9 and 10, are much larger. Consequently, the N values selected are typically smaller. In each of the following three chapters, the N values selected for the final models are discussed and justified in terms of data availability, feature size, and inference time tradeoffs.

7.2.4 Linear Probing

A linear probing strategy is introduced to aid model convergence and performance. In this technique, the non-linear layers of the model are initially frozen while only the linear layers are adjusted. This aids the model in extracting high-level features and can be particularly useful for training on small data sets. Additionally, it has been shown that models trained using a linear probing strategy can have higher out-of-distribution test accuracy than models trained with fine-tuning only [29]. In models trained through the scope of this work, a linear probing strategy is utilized in almost all cases. In these instances, the non-linear layers are frozen for a set number of epochs initially, before being unfrozen and fine-tuned.

7.2.5 Data Augmentation

Data augmentation helps prevent overfitting and increase model performance and is especially crucial when working with small data sets [30]. Several augmentation methods are applied in varying capacities to the different training sets and models gathered by our team. Within a given model, up to eight unique data augmentations are applied to each training and test image through Torchvision's transformations library [32]. One such augmentation is a color inversion. The rest of the augmentations are random flips and rotations about various axes. As a result, each data point is utilized to create up to seven unique data points and then appended to the training and testing data sets.

A second form of data augmentation employed in this study is the generation of synthetic data. Synthetic data creation entails the generation of completely new data points. The use of synthetic data

paired with real data has been shown to increase model performance, but comes at the risk of domain shift - a phenomenon in which the test accuracy on real data points falls [32]. This may occur in part to the model latching onto certain unintended artifacts that come with generated synthetic data. To overcome this, synthetic training data can also be created for the good training set in some cases [33]. Synthetic data generation is explored in the central defects model discussed in Chapter 10 given the especially sparse nature of defect data for center defects.

7.3 Model Performance Metrics

Several metrics are utilized to evaluate the performance of models trained in this work. These are accuracy, precision, recall, specificity, and F1 score. Each of these is a function of the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) a model identifies in a data set. A TP is when the model correctly identifies a defect, and a TN is when the model correctly classifies the image as not containing a defect. A FP is when the model wrongly predicts the image as containing a defect, and a FN is when the model does not identify a defect as being present even though there is one.

Accuracy refers to how correct the model is in overall classification. To calculate accuracy the following equation is utilized.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{7.4}$$

While useful, accuracy does not give insight to the relative ratios of FP and FN. This is crucial to understand as FP and FN have different implications. In our case, FP means the specialist is notified of a defect being present even though none exists. This is preferable to a FN, as in that case no notification occurs and the defect is missed causing scrap or yield loss. Precision is defined by the following equation.

$$Precision = \frac{TP}{TP+FP} \tag{7.5}$$

It is the ratio of correct defect identification to incorrect defect identification for the positives only. This is useful in isolating the cases the model predicts positives only. Recall is the ratio of correct defect identification to that plus the times the model misses a defect. It is described by the equation below.

$$Recall = \frac{TP}{TP+FN} \tag{7.6}$$

Recall is a key metric for models trained throughout this study as FN must be minimized to ensure that defects do not make it through the inspection module. Thus, recall should be maximized.

Specificity is the ratio of correct good wafer identification to that plus times the model wrongly identifies a defect as being present. Its equation is shown below.

$$Specificity = \frac{TN}{TN+FP} \tag{7.7}$$

This gives insight into how well the model passes truly negative samples relative to its false positive rate, mirroring the recall metric. Lastly, the F1 score is the harmonic mean of precision and recall. Its formula is below.

$$F1\ Score = \frac{2*Precision*Recall}{Precision+Recall} \tag{7.8}$$

Typically, precision and recall are difficult to maximize together as models slant towards too many FP or FN. F1 score gives insight into the balance of these two metrics. No one metric is utilized to fully evaluate an ML model. Rather, the scores are analyzed in union to understand model tendencies across data sets.

7.4 Localization Heat Map Activation

Gradient-weighted class activation mapping (GradCAM) is utilized to spatially analyze activations at the convolutional layer of the network in inference. GradCAM is a form of weakly supervised localization that creates visualizations to aid in class discrimination and the identification of biases present in data sets [34]. Through its implementation, the areas of an image that contribute most to the model's decision become visible. These highlighted areas should align with where defects are present in a well-functioning model. When paired with the model's confidence for a given decision and metrics from Section 7.3 across test data sets, greater understanding can be garnered.

7.5 Receiver Operator Characteristic Curves

Receiver operator characteristic curves help illustrate the capabilities and performance of a binary classifier, such as the models trained in the scope of this work. These curves are generated based on the true positive rate (TPR) and false positive rate (FPR) of a model. Within the model's classification, different threshold values are set for being considered a positive based on the confidences outputted by the model. For each of these thresholds, the TPR and FPR are recorded and subsequently plotted to create

a ROC curve. Figure 7-2 below shows examples of ROC curves with accompanying area under the curve (AUC) scores.

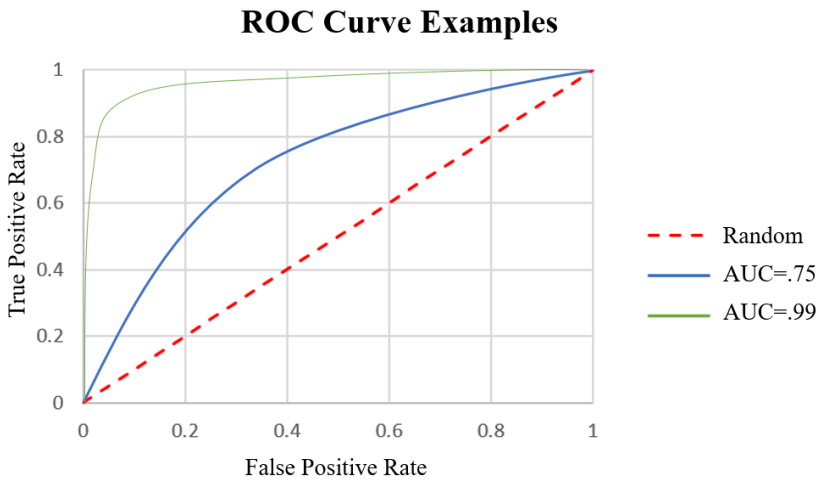


Figure 7-2: Receiver Operator Characteristic example curves: three ROC curves are shown above for an AUC of .5 (random), AUC of .75, and AUC of .99. Higher AUC values indicate a better model.

In the figure above, the dashed red line corresponds to random guessing. In contrast, the blue line is a model trained to recognize defects with an AUC of .75, and the green line corresponds to an AUC of .99. In the case of random guessing, the TPR and FPR are always equal to one another. For a model with better or worse than random guessing accuracy, the curves become less linear. From here, the optimal operating threshold in terms of TPR and FPR can be selected for a model and then used for inference. These curves can be generated based on distribution or out-of-distribution data. For more robust inference, out-of-distribution data must be considered.

7.6 Chapter Summary

In this chapter, the utilized machine learning architecture, functions, parameters, and methods were introduced. Metrics for model evaluation were also described, including scores like accuracy, precision, f1, and recall. Concepts used to understand model performance, such as GradCAM and ROC curves, were also detailed.

Chapter 8

Spin on Glass Defect Detection Model

In this chapter, only a brief summary of the model for detecting SOG defects is provided. For an in-depth discussion of this model, see Sampson [6].

8.1 Summary of SOG Model Motivation and Methods

SOG defects are non-reworkable defects that occur outside the photolithography module. Although non-reworkable, it is crucial these defects are caught fast. This is because they are often the result of systematic tool variation affecting all wafers processed by a tool. So, the ability to correctly identify a SOG defect early allows the team to prevent large containment events in which hundreds of wafers are affected. Also, rather than direct integration to the user interface, the SOG model is better used as an out of line engineering tool that continuously monitors wafers run through SOG tools at logpoints the defects may be visible. This is because SOG defects are difficult to see due to their size and coloring. As a result, a large N value is required for ML training, so that the features take up a larger percentage of the image trained on.

SOG data is sourced entirely from engineering contaminants and trained on N=8 and N=12 data splits with the architectures and parameters described initially in Chapter 7. Initial models were trained on a single SOG containment event with approximately 500 data points after an N=12 split. Results were insufficient and overfit. While working, a second SOG containment was discovered leading to an explosion of available data. Figure 8-1 shows examples of SOG defects from the improved data set.

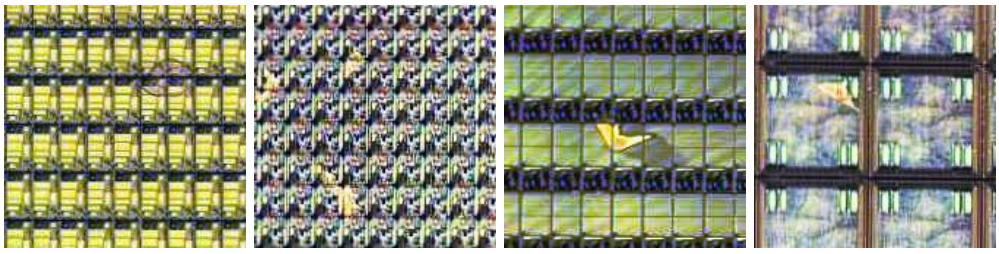


Figure 8-1: SOG defect examples. Examples of SOG defects at an N=12 split on different technologies.

The above SOG defect samples exist over a far greater number of layers and technologies than the initial containment data. Additionally, far more wafers are affected in the new containment, leading to even more data. So, data from the new containment is utilized for training, validation, and distribution testing, whereas the initial containment data is withheld for out-of-distribution testing. Table 8-1 shows the breakdown of training, validation, and testing data for new models trained. Additionally, all data points are color equalized initially.

Table 8-1: The sizes of the new data set for SOG defects before augmentation [6].

Data Split	Training Data	Validation Data	Test Data - In Distribution	Test Data - Out-of-Distribution
N = 12	4512 images	1128 images	1410 images	828 images

With the above split, two model configurations are trained with a cross-validation strategy. Both models utilize eight augmentations and pre-trained weights and biases. The first model presented in this work is configuration 7, which is trained with no linear probing. The second is configuration 8, which is trained with linear probing. Within each configuration, a weight decay of 0.001 is used in one version, and not in the other.

8.2 Summary of SOG Model Key Results

Table 8-2 below shows the test accuracy, F1 score, and AUC score for each of the configurations on both in-distribution and out-of-distribution data. For both configurations 7 and 8, results are superior for models not using weight decay. This indicates the weight decay parameter of 0.001 selected may be too high, or even unnecessary given the large data set utilized for training. Configuration 8 with no weight decay is the superior model across the four trained, marking the highest accuracy metrics across all six determined excluding the out-of-distribution F1 score. The out-of-distribution AUC achieved for configuration 8 with no weight decay is 0.927. The accompanying in and out-of-distribution ROC curves are shown in Figure 8-2.

Table 8-2: Accuracy metrics for configurations 7 and 8, without and with weight decay [6].

Configuration	Test Accuracy		Test F1 Score		AUC Score	
	In Distribution	Out-of-Distribution	In Distribution	Out-of-Distribution	In Distribution	Out-of-Distribution
7 (N = 12, without weight decay)	89.6%	86.4%	0.895	0.866	0.961	0.914
7 (N = 12, with weight decay)	77.4%	51.3%	0.792	0.439	0.860	0.559
8 (N = 12, without weight decay)	91.6%	86.9%	0.917	0.855	0.964	0.927
8 (N = 12, with weight decay)	86.2%	68.4%	0.863	0.679	0.927	0.749

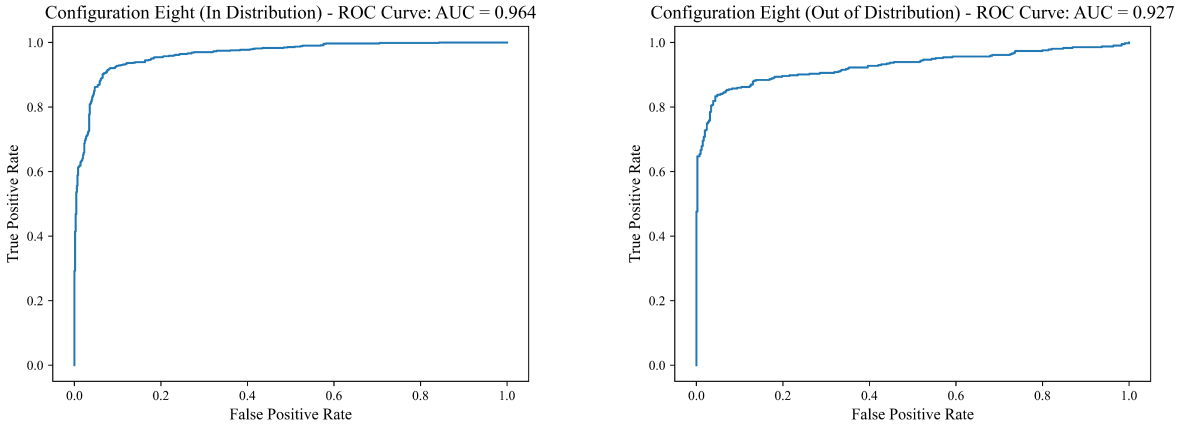
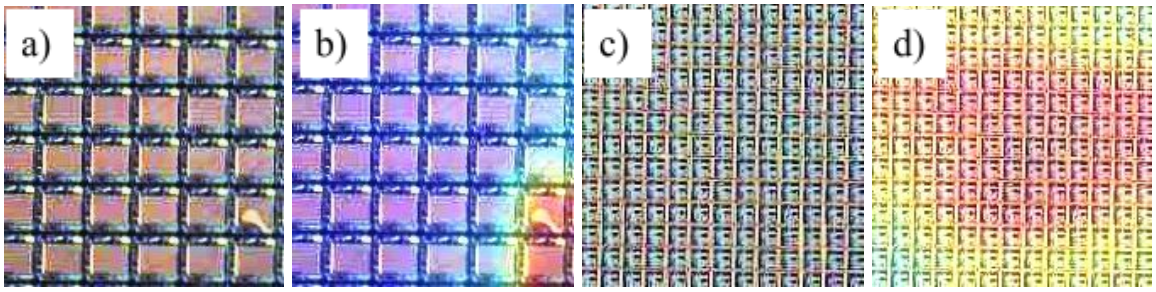


Figure 8-2: In and out-of-distribution ROC curves for configuration 8, no weight decay.

GradCAM is applied to the out-of-distribution data run on configuration 8 with no weight decay. Figures 8-3a and 8-3b show an example of a true positive and the associated activations. Activations are localized to the area of the wafer containing the SOG defect. Figures 8-3c and 8-3d show activations for a true negative case, where no areas are localized. This is indicative of the model's ability to abstract SOG defects and make the proper classifications. While these cases are promising, and constitute a majority of

inferences; false positives and negatives both exist. In these cases localization often occurs needlessly to artifacts of different wafer technologies, or the defects are localized but misclassified. An increase in training data should help with this issue.



Figures 8-3a, 8-3b, 8-3c, and 8-3d: a) Configuration 8 true positive from out-of-distribution, b) GradCAM of 8-3b, c) Configuration 8 true negative from out-of-distribution set, d) GradCAM of 8-3c. The highest activation areas are red.

8.3 Summary of SOG Model Future Work

Several models are trained for detecting SOG defects on wafers for deployment in an engineering tool outside of the OwlView interface. The best performing model uses linear probing and no weight decay to reach an out-of-distribution AUC of 0.927. As more SOG data becomes available, it should be fed into the model to enhance accuracy metrics and most importantly the out-of-distribution performance.

Chapter 9

Edge Defect Detection Model

In this chapter, the model developed for detecting edge defects is discussed only at a high-level. For an in-depth discussion of this model, see Cheung [35].

9.1 Summary of Edge Model Motivation and Methods

Edge defects are the most common reworkable defects linked to the photolithography process. They include wide EBR, eyelash, and spin defects as shown in Figure 9-1. Each of these is the result of systematic variations in a pump's ability to dispense a photocurable resin or systematic variations in the wafer's surface. Thus, a binary classifier model is trained for the identification of defects along the edge of a wafer.

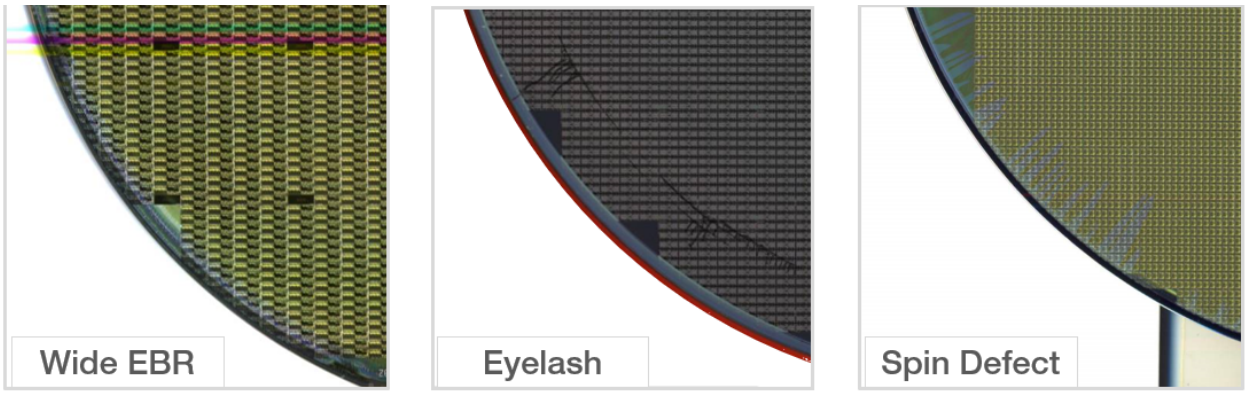


Figure 9-1: Examples of wide EBR (left), eyelash (center), and spin (right) defects in an N = 3 split [35].

Data for this defect class is relatively abundant, with over 1500 whole wafer images available containing spin defects of one sort or another. Splits of N=3 and N=5 are selected for investigation as spin defects are comfortably visible in these image splits, and real-time inference is required for this model. As a result, a small N lends to fewer inferences per wafer and per lot - leading to the consumption of less computational power. After applying a single color inversion augmentation to both the N=3 and N=5 data sets, 48,000 and 96,000 total images are available for training in each of the cases, respectively. Additionally, this model is trained on only color-equalized images just like all other models, as defects are

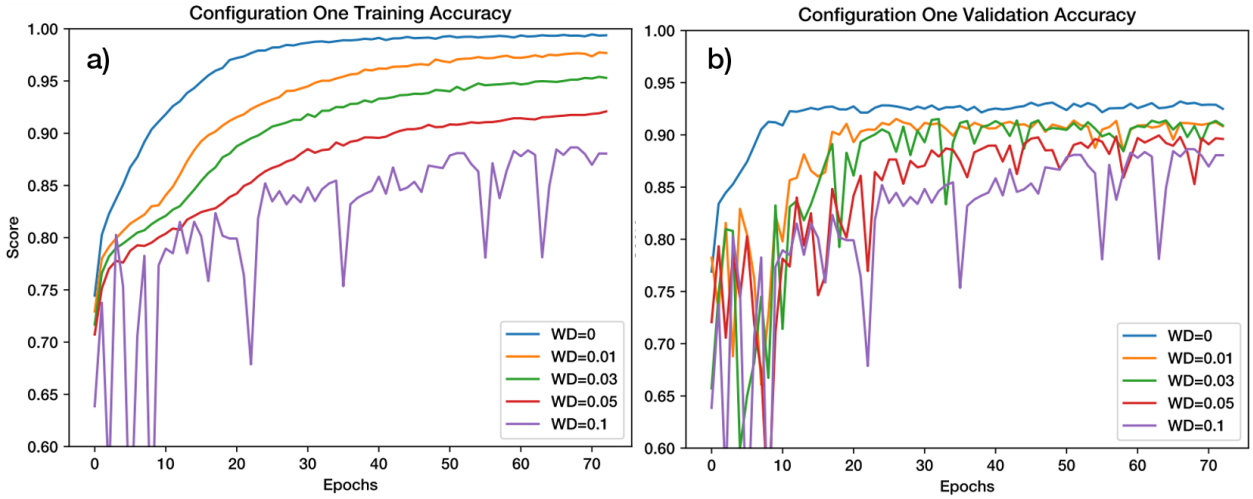
more visible. Table 9-1 shows the configurations of various models trained for the edge model. Varying weight decays are applied and experimented with to aid in model convergence and out-of-distribution performance.

Table 9.1: Configurations tested for the edge defect learning model [35].

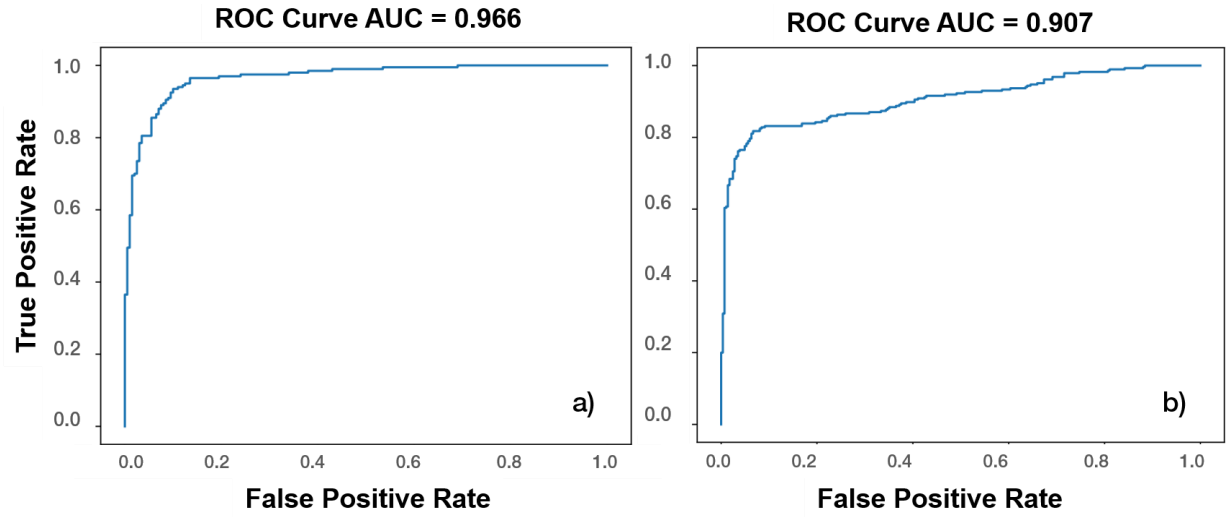
Center Defect Configuration Matrix							
Configuration	**N**	**Linear Probe**	**Weight Decay**				
1	3	Off	-	0.01	0.03	0.05	0.1
2	3	On	-	0.01	-	-	-
3	5	Off	-	0.01	-	-	0.1
4	5	On	0.001	0.01	-	-	-

9.2 Summary of Edge Model Key Results

Of all models trained, configuration 1 displayed superior results. Figures 9-2a and 9-2b show a single fold's training and validation accuracies across varying weight decays. Further, ROC curves for the in and out-of-distribution test sets are shown in Figures 9-3a and 9-3b. While the model with no weight decay does achieve a higher validation accuracy than other model iterations, its out-of-distribution performance is inferior to the model with a weight decay of 0.01.

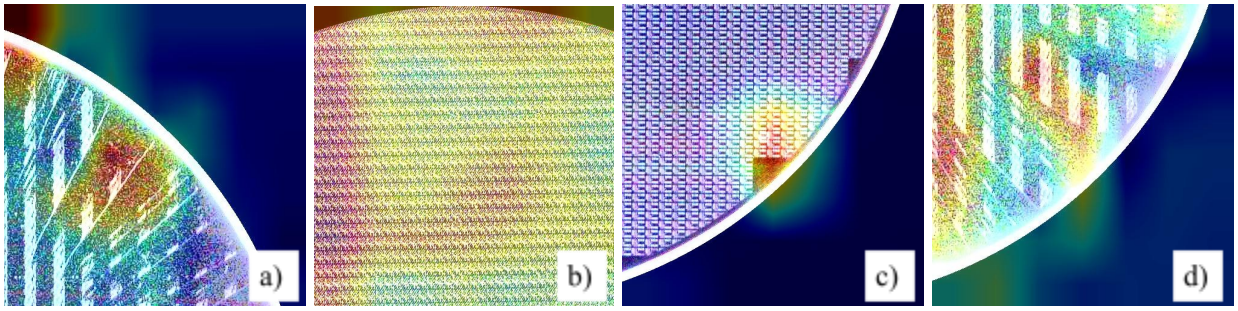


Figures 9-2a and 9-2b: Configuration 1 results across five different weight decays for [35] a) training accuracies, and b) validation accuracies.



Figures 9-3a and 9-3b: AUC scores and ROC curves plotted for the Configuration 1 model with weight decay = 0.01 tested on a) test set in distribution, and b) test set out-of-distribution.

GradCAM is applied to the out-of-distribution data run on configuration 1 with a weight decay of 0.01. Figure 9-4a shows an example of a true positive, where activations appear to correctly localize spin defects on the edge of a wafer. Figure 9-4b shows an example of a true negative, where activation is more diffuse, indicating no false edge defect localization by the model. Figure 9-4c shows an example of a false positive where the model wrongly localizes a feature of the wafer technology. Figure 9-4d shows an example of a false negative, in which spin defects appear to be localized by the model but the wrong classification is made. Each of the false cases are expected to occur with reduced frequency as more data is fed into the models.



Figures 9-4a, 9-4b, 9-4c, and 9-4d: GradCAM images for edge defects [35]. a) true positive, b) true negative, c) false negative, d) and false negative

9.3 Summary of Edge Model Future Work

Results of the edge models show promise and some production readiness - with an out-of-distribution AUC approaching 0.91. The models trained show the ability to localize and correctly classify edge defects but could be improved with increased data. As the OwlView interface is implemented on the fab floor, the availability of data is expected to grow - and model performance will improve subsequently.

Chapter 10

Center Defect Detection Model

In this chapter, the model developed for detecting center defects is discussed in detail. Methods and techniques utilized to overcome the extremely sparse nature of the data- such as the generation of synthetic data - are described. Models trained are outlined in detail.

10.1 Problem of Data Availability for Central Defects

The central model is the last model pursued by our team, partly due to the unique challenges presented by data availability. A wide array of defects may affect the center of any given wafer. These include but are not limited to centerholes, particles, hotspots, flashfields, contaminations, and scratches. All of these defects have unique visual qualities. Examples of each of these are shown in Figure 10-1 below.

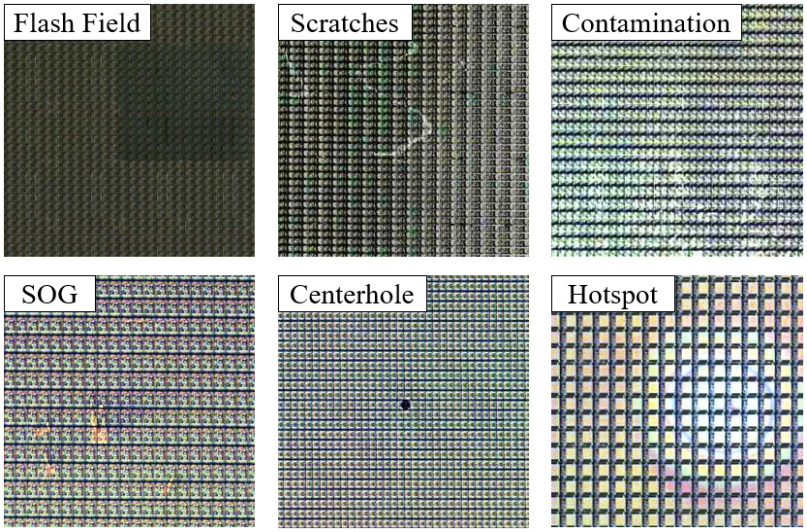


Figure 10-1: Examples of different center defects. The above images are taken from N=5 splits of wafer images tagged by the team. These examples show different underlying technologies for each defect type.

The above figure shows a single example of each of the defect types, but the shape, coloring and sizes of the defects can vary greatly. Figure 10-2 illustrates this by presenting a sample of different hotspot defects.

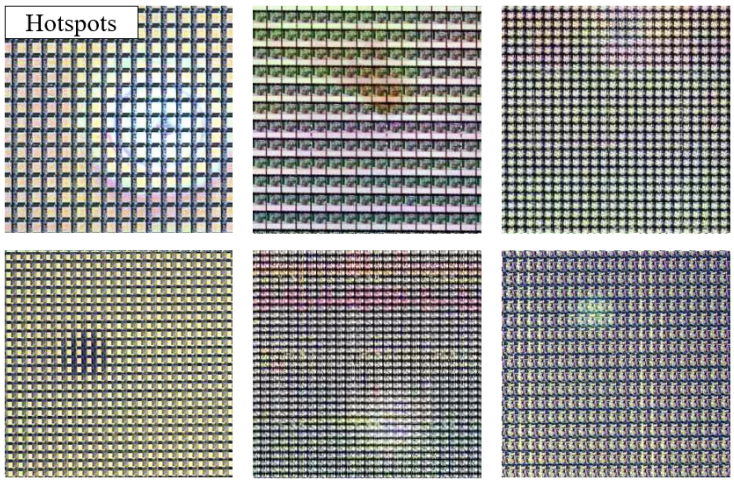


Figure 10-2: Examples of different hotspots. The above images are taken from N=5 splits of wafer images tagged by the team. This figure displays the array of ways in which a hotspot might appear.

To further complicate this, along with each layer of the wafer looking different between logpoints as shown previously in Figure 2-3, the underlying technology of the wafer can vary greatly as DFAB produces over 1000 technologies, with six disparate examples shown below in Figure 10-3.

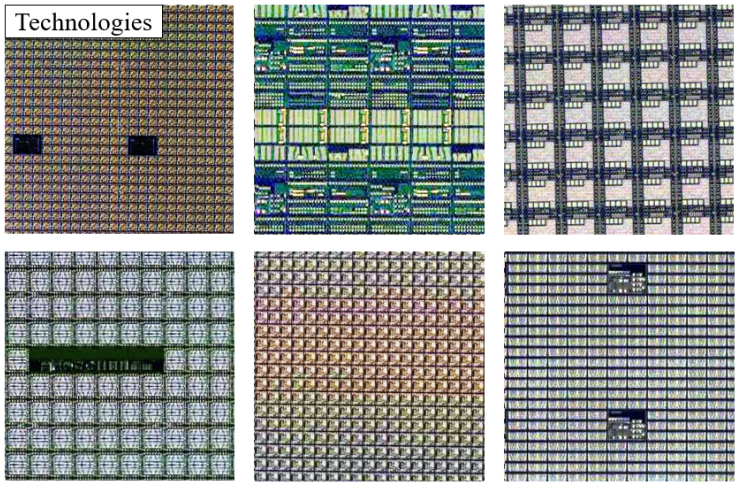


Figure 10-3: Examples of different technologies. The above images are taken from N=5 splits of wafer images tagged by the team. No defects are present on these wafer sections.

For the SOG model discussed in Chapter 8, many recent engineering contaminants are available, which made it relatively easy to source and tag data. Additionally, SOG defects are only visible on a handful of logpoints, so concerns about the model's applications to other layer levels are not as relevant. For the spin defects model discussed in Chapter 9, rework data is widely available on online databases

making the creation of a more robust model feasible. For central defects, this is not the case. Data is less available, and the man hours it takes to source a wide array of central defects across the hundreds of technologies and dozens of logpoints made an approach mirroring those taken in Chapter 8 or 9 impractical.

10.2 Methodology for Central Defects Model

In this subchapter, the model parameters for the four central defect models trained are first described. Next, model evaluation criteria are discussed. Finally, methods for generating synthetic data as utilized in the models are outlined.

10.2.1 Proposed Experiments for Central Defects Model

The team proposes that by augmenting our data set with synthetic data for different central defect types, we can create a model capable of detecting different central defects across layers and technologies not represented in the training set. To achieve this, hotspot defects are selected as the single defect class to focus on initially to prove the validity of the synthetic data creation strategy. This is justified for three reasons. First, hotspots are the most abundant in our data set. After tagging, the team collected 700 unique hotspot defect occurrences. While relatively ample, many of these defects seem identical. This is because hotspot defects often affect a whole lot of wafers, rather than just a single one. Figure 10-4 shows examples of four unique hotspot defects affecting different wafers within the same lot.

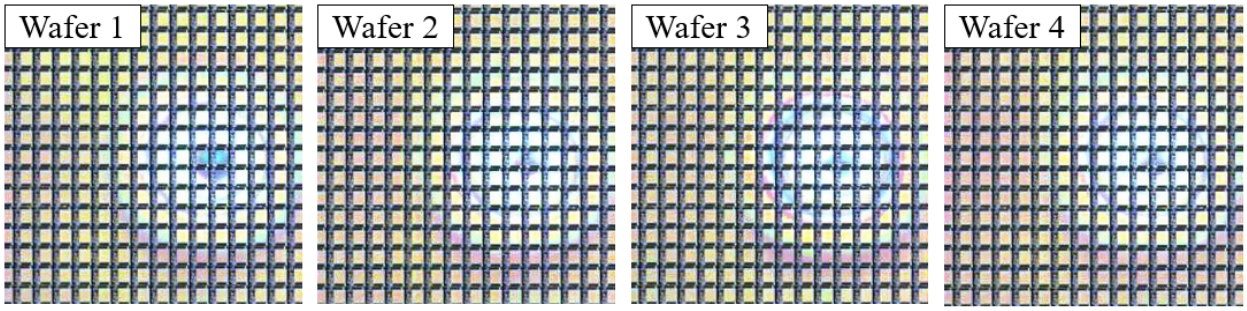


Figure 10-4: Examples of the same lot, different hotspots: four unique wafers within the same lot have similar-looking hotspots. These hotspots also appear in the same location on each wafer.

The above figure helps illustrate the second motivation for selecting hotspot defects. They are often missed by EV's rules-based detection algorithm because they affect every wafer within a lot. Consequently, the golden wafer and all other wafers look similar and detection accuracy suffers. The

above figure also helps demonstrate that, while 700 unique hotspots exist, only around 50 are completely unique in visual properties - further justifying the need for a synthetic data approach to be applicable across layer levels and technologies. The third motivation in selecting hotspots over other defects is their distinct patterning. While hotspots come in many shapes and sizes, through various imaging editing techniques, convincing synthetic ones are creatable through various image modification strategies.

So, to establish the feasibility of this technique, models with the data sets described in Table 6-1 are trained. For each model, an N value of five is utilized along with pre-trained weights and biases. Additionally, ten epochs of linear probe, the Adam optimizer with an initial learning rate of 1E-4, and eight data augmentations are utilized.

Table 10-1: Data set configurations for hotspot binary classifier.

	Number of Good Images			**Number of Bad (Hotspot Defects)**		
Config:	**Real**	**Synthetic One Goods**	**Synthetic Two Goods**	**Real**	**Synthetic One Defects**	**Synthetic Two Defects**
1	576	0	0	576	0	0
2	576	576	0	576	576	0
3	576	576	576	576	576	576

In the above table, parameters and data inputs for the models are described for three unique configurations. In the first configuration, there is no synthetic data. In the second configuration, there is synthetic data generated through the use of our first method (called synthetic one) for creating said data. In the third configuration, there is data created using both the first and second methods (synthetic two) of synthetic data generation.

A fourth model configuration is also devised for detecting all types of central defects with the same optimizer and learning rate. This model has significant limitations regarding the availability of real defect data across defect types. For centerholes, only 213 unique defect images exist, with some of those being repeats affecting a whole lot. For contaminations, there are only 129 unique defect images gathered through the scope of our work. For scratches, there are only 146 unique defect images. In each of these cases, the defect images are already subdivided using an N=5 split.

While a classifier model could be trained on this limited data set, it would be overfit and perform poorly on out-of-distribution data. Additionally, there would be so little out-of-distribution data to test. Instead, a binary classifier model is devised for configuration 4as shown in Table 6-2.

Table 10-2: Data set configuration for central defect binary classifier.

Configuration 4						
Good Images		Bad Images				
Real	Synthetic	Real Hotspots	Synthetic Hotspots	Synthetic Contaminations	Synthetic Scratches	Synthetic Centerholes
1728	1728	576	1152	576	576	576

In the above configuration, the model is trained on a one-to-two ratio of real to synthetic hotspots, exactly identical to configuration 3. In addition, the model is trained on only synthetic centerholes, scratches, and contaminations. The aim of this model is to evaluate performance on out-of-distribution real defects.

For each of the described models, parameters like N, use of pre-trained weights and biases, and learning rates are used based on finding from the models trained in Chapters 8 and 9. For a more in-depth discussion of those parameters, refer to those sections. While additional adjustment of said parameters can impact and improve model performance, the chief motivation of the central defects model is proving the viability of a synthetic data creation strategy to create models applicable across the many technologies and logpoint types at DFAB. Consequently, model hyperparameters are held constant. In future iterations of these models, these can be further tuned. Within models, the weight decay parameter is tuned.

10.2.2 Evaluation of Models

To garner an understanding of the effects of the synthetic data's inclusion in the data set, the models are trained using a cross-validation strategy as described in Chapter 7. Here, the good and bad data sets are split into training and test sets with respective ratios of 80% to 20%. For the training set, a k-fold of 5 is utilized such that five unique models are trained with each training set split into 1 of the 5 unique validation sets respectively. The remaining test set is then utilized to inference and evaluate the models' performances on data it never saw throughout training. In this case, the defects and layer levels are likely represented in the training data set. This is because the test set is randomly pulled from the total good and bad splits. While useful, we aim to evaluate the model's performance on unseen data not represented in the test set.

To achieve this, a second test data set is created containing real hotspots from layer levels and technologies completely unseen by the model. For configurations 1 to 3, this dataset contains 115 good and 115 bad examples of real hotspots. This data set is utilized to evaluate the model's performance on technologies and hotspots with characteristics the model has never seen before. For the fourth

configuration model encompassing all center defects and synthetic data, 124 real defects of each type are withheld for out-of-distribution inferencing.

10.2.3 Generation of Synthetic Data

To generate synthetic data, images of good wafers are first required. It is important these good wafers are from a wide array of technologies and layer levels. To achieve this, an SQL query is utilized to randomly pull lots and logpoints from the different technology groups produced by DFAB in an even distribution. While powerful, this data still requires tagging so that no defective wafers are included in the good data set. In total, for the first three configurations on hotspots only, 188 unique lots of wafers are identified and tagged. For the fourth configuration only, 375 additional lots are added to increase the number of technologies included in the data set. In each of these lots several logpoints are pulled as well.

These images are split and sorted with the methodologies described in Chapter 5 of this work. From here, two primary synthetic data generation methodologies are applied to each wafer image. These are “synthetic one” and “synthetic two.” Synthetic one is applied first in z configuration 2. Synthetic two is created next for configuration 3, which contains data from both methods.

For the synthetic one, a Python script is created that reads in a good wafer image, filters out dark pixels that outline the individual die, applies a hotspot-shaped mask to the remaining underlying area, modifies the masked area with brightness and saturation changes, and then recombines the images such that the hotspot appears underneath the rigid geometric patterns as typical in reality. Figure 10-5 displays this methodology.

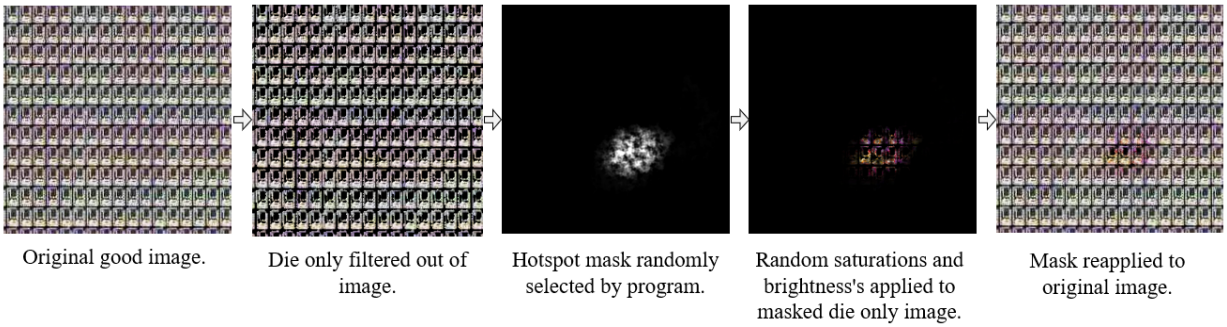


Figure 10-5: Synthetic data generation process example: in the above procedure, the mask shape and spatial properties are randomized as well.

In each of these steps, elements of randomization are deployed so that no two hotspots generated look the same. The mask that eventually becomes the hotspot is randomly selected from a folder

containing 50 hand-generated masks using different brushing techniques in Microsoft Paint. These masks are randomly flipped and altered such that an array of positions and sizes are present. The brightness and saturation factors are also randomly generated for each image.

The method was arrived upon after much investigation and experimentation. Initially, a solid-colored mask was simply placed on top of the wafer image. After training and evaluating those models' performances, it was clear that the model was simply discovering whether or not the masked layer was present - as it caught no real defects in inference. The new method shows promise for two reasons. First, defects generated are embedded within the actual image rather than simply being the last mask level. Second, because parameters like saturation and brightness are modified, the masked hotspots have color profiles with many varying colors. This stands in contrast to the earlier attempts that create uniformly colored pixels within each hotspot mask. Figure 10-6a shows examples of synthetic hotspots generated with synthetic one, whereas Figure 10-6b shows examples of real hotspots.

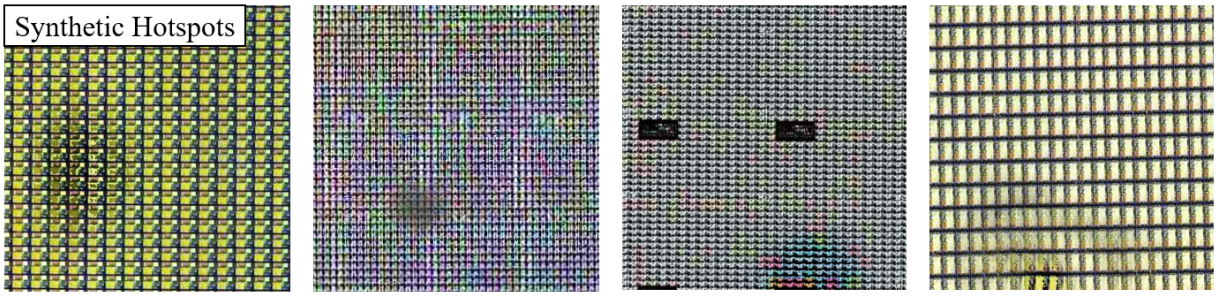


Figure 10-6a: Synthetic hotspot samples: hotspots generated using synthetic one.

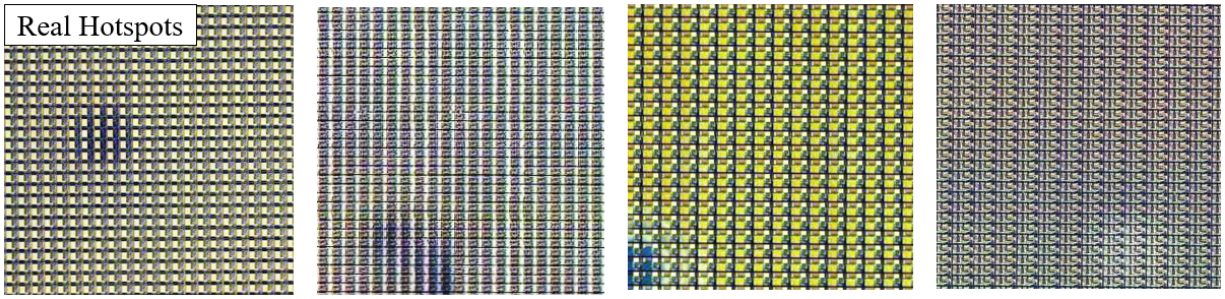


Figure 10-6b: Real hotspot samples: real hotspots gathered, tagged, and split by team.

Additionally, synthetic data is created for the goods to work against the models' ability to detect images modified by us. Here, the masks used are uniform across the whole wafer with varying opacities. With this script, thousands of synthetic data points are generated and then randomly downsampled to 576 for good and bad to match the data set for configuration one. This way, configuration two has an even balance of synthetic and real data points for hotspots.

The second method for generating synthetic hotspot defects works similarly. Like the first, a Python script is created that reads in a good wafer image, filters out dark pixels that outline the individual dies and applies a hotspot-shaped mask to the remaining underlying area. Here, rather than modifying the brightness and saturation, the color profiles are modified. Next, everything is recombined the same way, but the opacity of the color mask is randomly modified. This leads to authentic looking fake hotspots. Figure 10-7a shows examples of synthetic hotspots generated with this method, whereas Figure 10-7b shows examples of real ones.

Figure 10-7a: Synthetic hotspot samples: hotspots generated using synthetic two.

Figure 10-7b: Real hotspot samples two: real hotspots gathered, tagged, and split by team.

Synthetic two is used to create data for configuration 3. These methods are utilized to create synthetic hotspot data. For other defect types as explored in the fourth model configuration, different defect types are generated using the same methodologies. Here, only the mask geometries and ranges of random parameters like brightness, saturation, and color are modified. Figure 10-8a shows examples of all types of synthetic defects generated with this method, whereas Figure 10-8b shows examples of real ones.

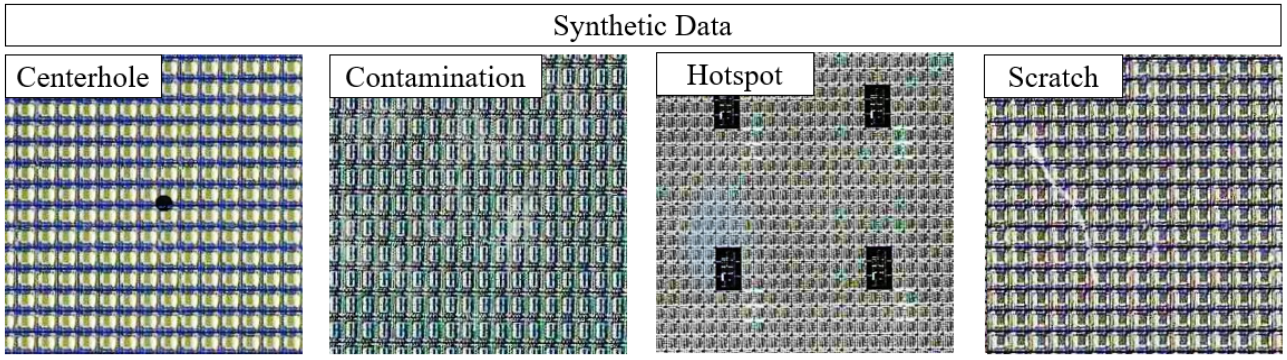


Figure 10-8a: Synthetic data samples all types. Examples of synthetic data generated for centerholes, contaminations, hotspots, and scratches. Many thousands of these are generated for each across layer levels and technologies. Additionally, the mask geometries and positions are randomized as well.

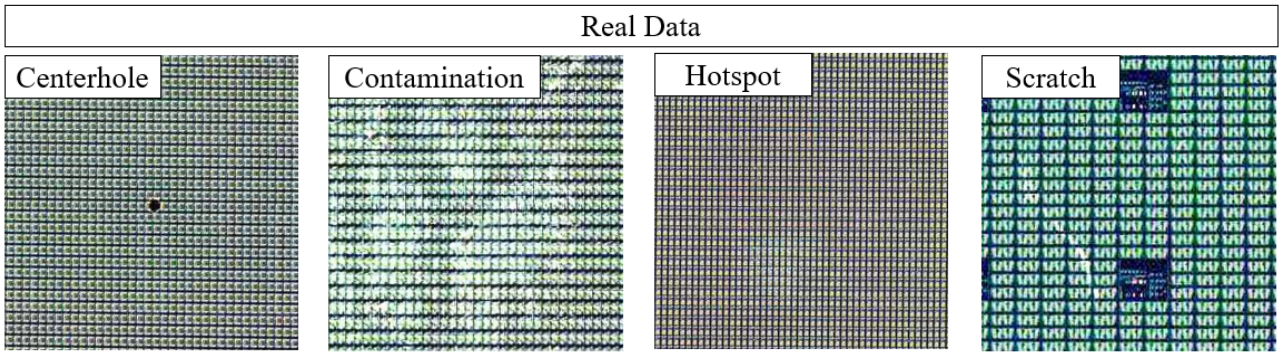
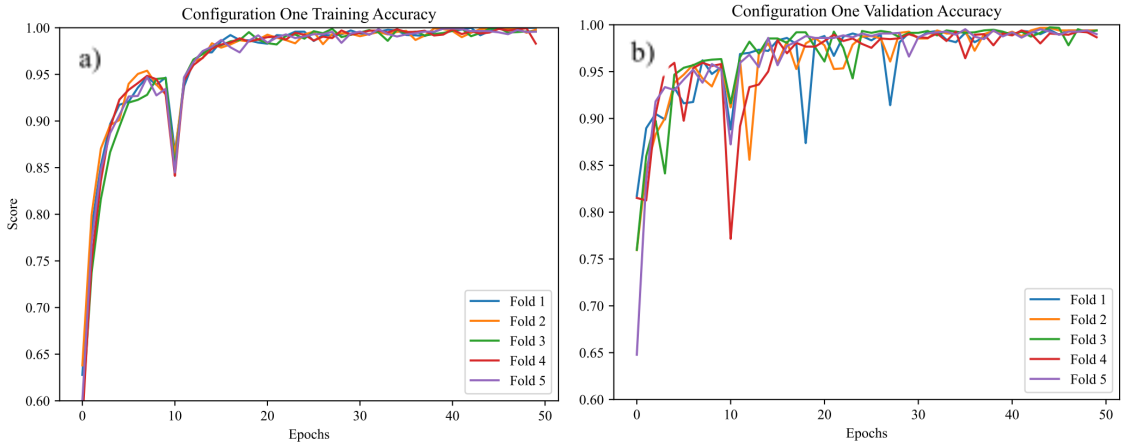


Figure 10-8b: Real data samples of all types. Examples of real data gathered for centerholes, contaminations, hotspots, and scratches.

The synthetic defects generated for these examples are randomly generated across over 300 lots pulled to capture a random distribution of the technologies produced by DFAB.

10.3 Model Results

Figures 10-9a and 10-9b show the training and validation accuracy scores of each of the five folds of configuration one, which was trained on only real hotspot data. Figure 10-10 shows the validation loss for each of the five folds of configuration one. Figure 10-11a shows the ROC curve for configuration 1 performance on the test data set - which is in-distribution data. Figure 10-11b shows the ROC curve for configuration one performance on out-of-distribution data.



Figures 10-9a and 10-9b: Configuration 1 training and validation accuracy across five folds. a) Training accuracy; b) Validation accuracy.

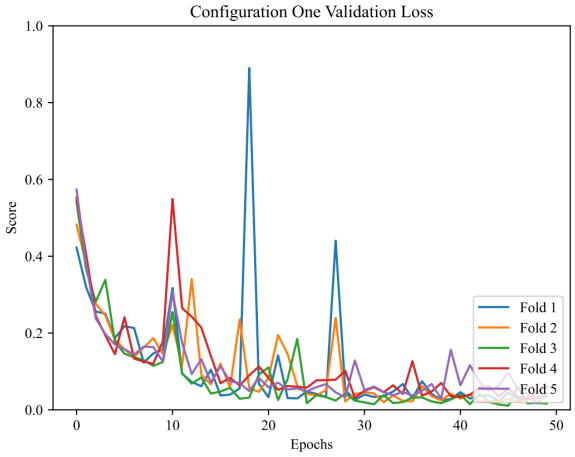
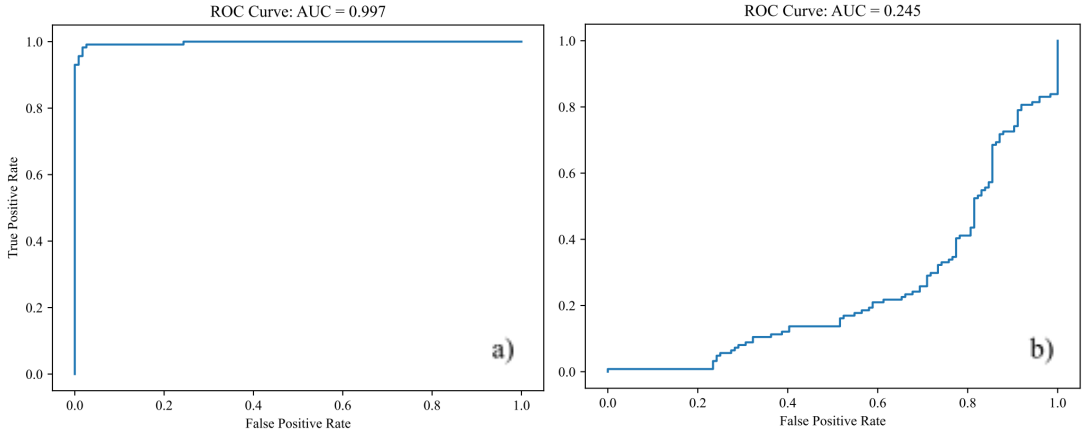
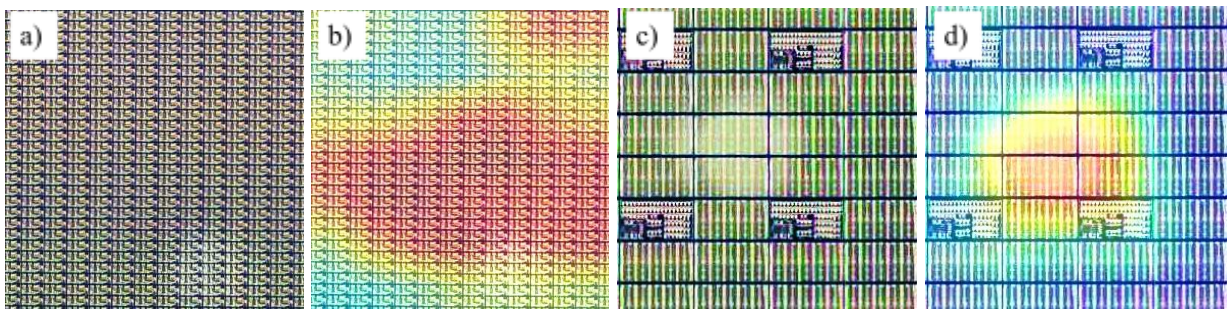


Figure 10-10: Configuration 1 validation loss. Validation loss plotted over 50 epochs for five folds.



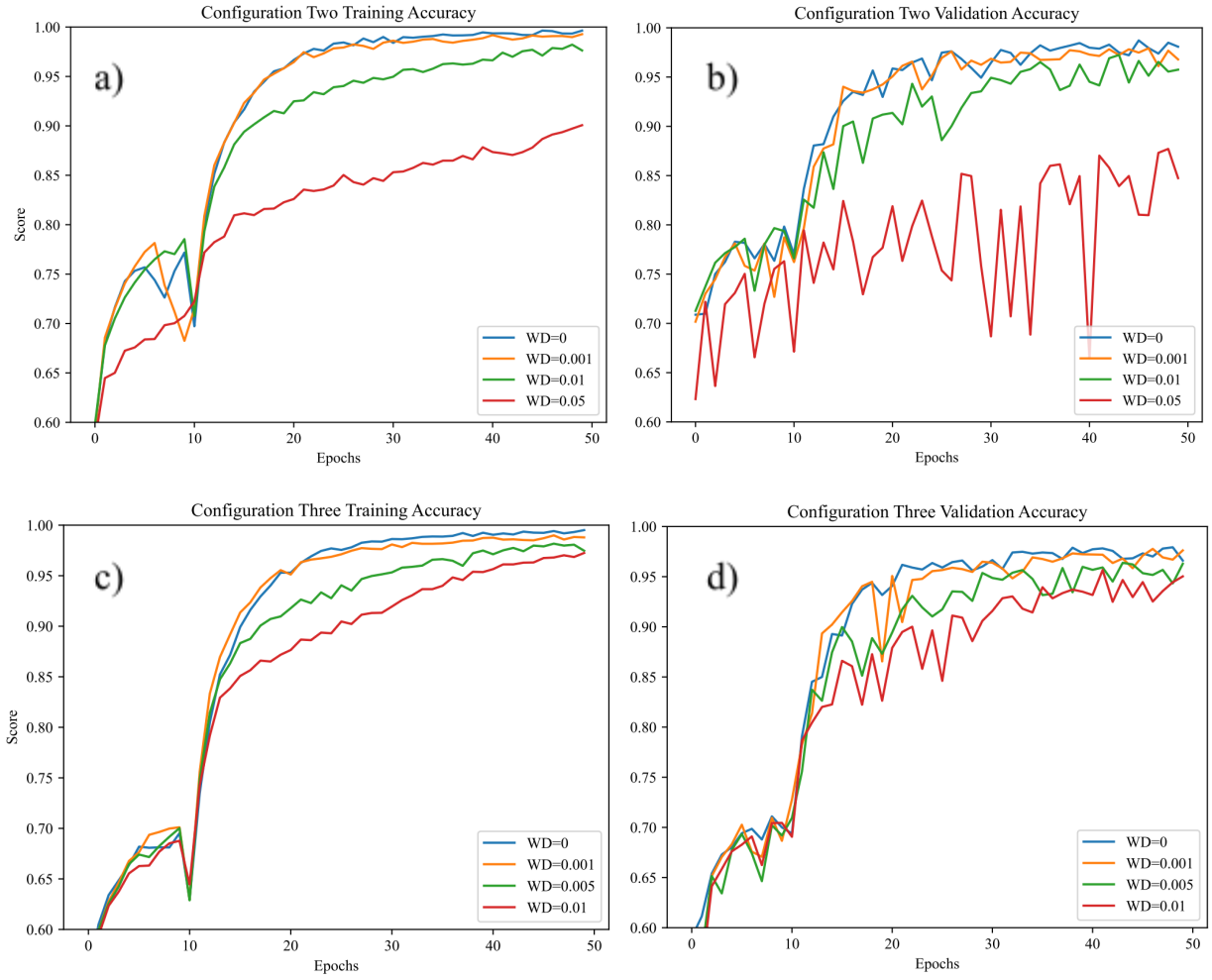
Figures 10-11a and 10-11b: Configuration 1 ROC curves: a) in distribution test set AUC=0.997; b) out-of-distribution data AUC=0.245.

The training and validation accuracies for configuration 1 converged to scores over 97.5% within 30 epochs as the validation losses fell. This model is grossly overfit, however, as out-of-distribution AUC performs worse than guessing odds. So while the model learned something, its ability to correctly classify out-of-distribution data is non-existent. These results are expected, however, given the limited nature of hotspot defect data, and the limited amount of layers represented. Figures 10-12a, 10-12b, 10-12c, and 10-12d show examples of hotspot defects and the model's corresponding activation maps.



Figures 10-12a, 10-12b, 10-12c, and 10-12d: a) Configuration 1 true positive from test data set (in distribution), b) GradCAM of 10-12a, c) Configuration 1 false negative from out-of-distribution det, d) GradCAM of 10-12c. The highest activation areas are red.

Figure 10-12a shows a true positive from the test data set, which is in distribution data that the model did not see. In this case, the activations shown in Figure 10-12b, are not localized to the hotspot defect shown in the bottom right corner. So, the model correctly classified this hotspot, but the area containing the hotspot did not have a high activation. This trend is true for other hotspots correctly identified in this case, which had the AUC score of 0.997 from Figure 10-11a. This model simply learned that this layer level and technology pairing will likely contain a hotspot defect since other nearly identical hotspots were fed into this model in training. For Figures 10-12c and 10-12d, a false negative from the out-of-distribution test set and its corresponding activation image is shown as run through configuration 1's model for inference. Although it is misclassified, activation is high corresponding to the hotspot defect. This further confirms the notion of overfitting, as although the model does have some ability to abstract hotspot defects for technologies it has never seen, its classification is still wrong as it is overfit for the data in the training set. Figures 13-a and 13-b show the training and validation accuracy scores for the configuration 2 models, which introduce synthetic data to the training set. Figures 10-13c and 10-13d show the same for configuration 3 models, which utilize two synthetic data creation strategies. For these models, only one fold is plotted at varying weight decays. The weight decay is adjusted to affect overfitting.



Figures 10-13a, 10-13b, 10-13c, and 10-13d: a) Configuration 2 training accuracy across weight decays, b) Configuration 2 validation accuracy across weight decays, c) Configuration 3 training accuracy across weight decays, d) Configuration 3 validation accuracy across weight decays.

For each of the above figures, models are slower to converge, and they converge to lower overall training and validation accuracies as the weight decay parameter, WD, increases. Weight decays vary slightly between models as they are iteratively adjusted and evaluated. For configuration 2, the .05 weight decay case was allowed to run longer, converging above 95% after 120 epochs. For configuration three, the 0.05 weight decay model never converged. The above models have lower training and validation accuracies than the configuration 1 model, which is encouraging as the configuration 1 model is overfit and performs poorly on out-of-distribution data. ROC curves are generated for each of the models for test data (in-distribution) and out-of-distribution data. Table 10-3 shows the AUC scores for each of the models.

Table 10-3: Test and out-of-distribution data set AUC values.

Configuration	Weight Decay	Test Data Set (in distribution) AUC	Out-of-Distribution Data Set AUC
1	0	0.997	0.245
2	0	0.980	0.423
2	0.001	0.967	0.520
2	0.01	0.980	0.589
2	0.05	0.965	0.538
3	0	0.958	0.778
3	0.001	0.957	0.664
3	0.005	0.955	0.809
3	0.01	0.951	0.714
3	0.05	0.763	0.337

For each of the data sets, a small increase in weight decay shows some benefit in model performance on out-of-distribution data. Too much weight decay eventually lowers a given model's performance, however. This typically comes at the cost of in-distribution performance, indicating a reduction in overfitting. Between configurations 1, 2, and 3 for the no weight decay cases, there is an improvement in AUC as shown in Figure 10-14.

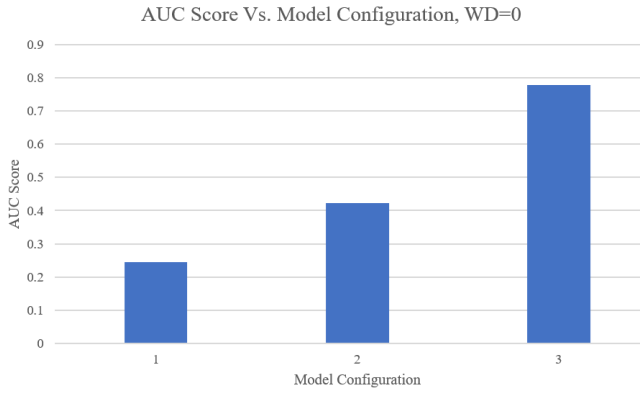


Figure 10-14: AUC vs. model configuration for weight decay of zero. AUC climbs between model configurations from 0.245 to 0.778 as synthetic data is introduced.

For the three model configurations above, everything is identical except the datasets. Configuration 1 has no synthetic data, 2 has synthetic data made with one method, and 3 has synthetic data made with two methods. Additionally, the amount of data doubles between 1 and 2, and triples between 1 and 3 as a result of the synthetic data. The large growth in AUC from 0.245 to 0.778 in the datasets without weight decay shows the viability of the synthetic data strategy in helping the model inference better on out-of-distribution data, which is key for this application. Of the model 3 configurations trained at varying weight decays, the highest AUC for out-of-distribution data is achieved with a weight decay of 0.005. This AUC value is 0.809; the corresponding ROC curve is shown below in Figure 10-15.

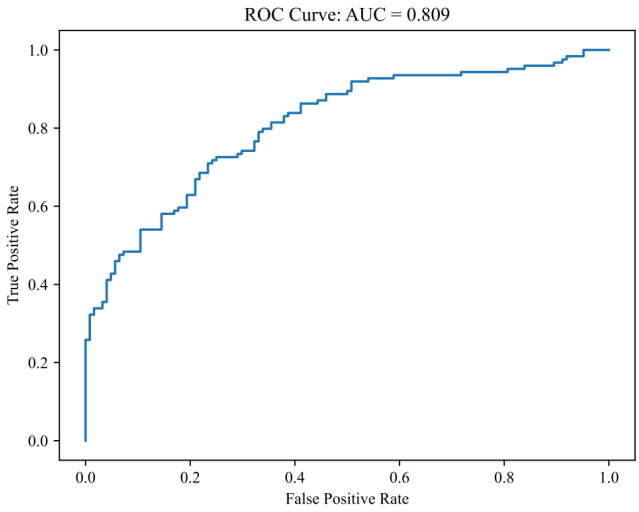
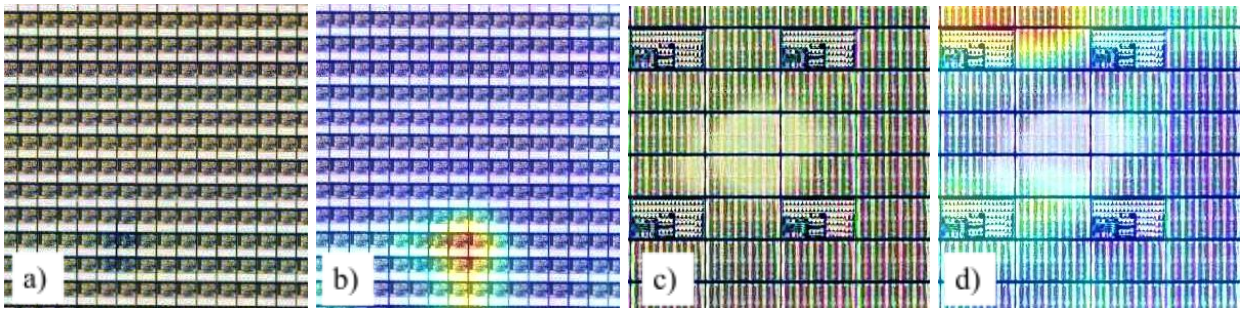


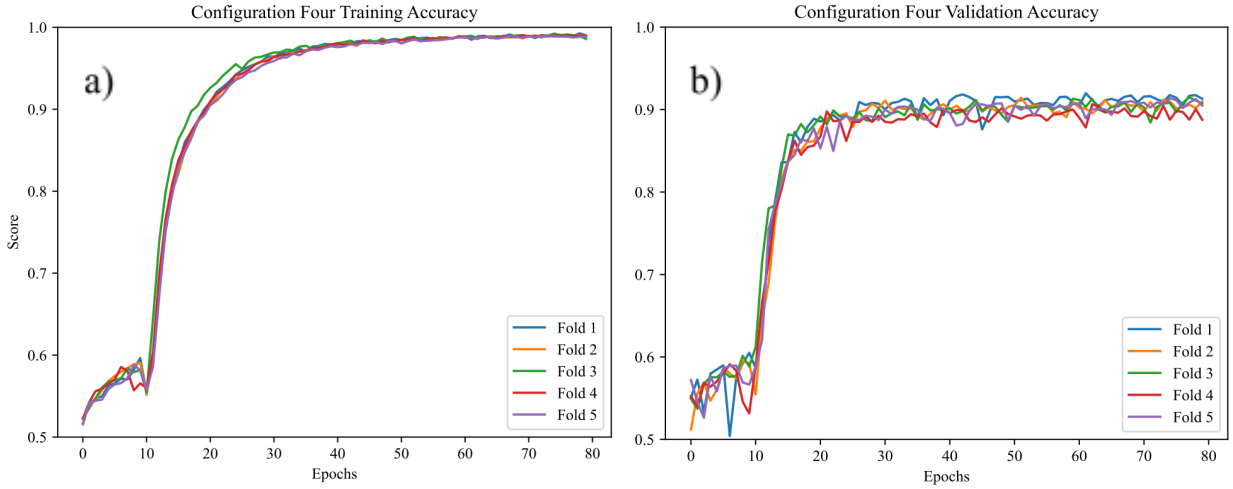
Figure 10-15: ROC curve for configuration 3 and weight decay of 0.05. Receiver operator characteristic curve for hotspot-only model with best AUC of models trained on out-of-distribution data.

Considering the extremely limited dataset of real hotspots, this is a promising result that is not quite production ready. Although the true positive rate is up to three times better than the false positive rate in certain operating points, the odds of false positives are still large, as nine inferences are required for every wafer. If the false positive rate is too high, the operator will likely ignore OwlView suggestions entirely. Thus, a higher AUC is desired for production. Integrating synthetic data strategies and weight decay to reduce overfitting led to a large increase in AUC from the original overfit models trained on only real hotspots. While true, a greater amount of data is simply required to utilize the central defect model. This data can be gained once the system is implemented on the fab floor. Examination of GradCAM images from model 3 gives some insight into its performance as seen in figures 10-16a through 10-16d.



Figures 10-16a, 10-16b, 10-16c, and 10-16d: a) Configuration 3 and weight decay of 0.05 out-of-distribution true positive example, b) GradCAM of 10-16a, c) Configuration 3 and weight decay of 0.05 out-of-distribution false negative example, d) GradCAM of 10-16c.

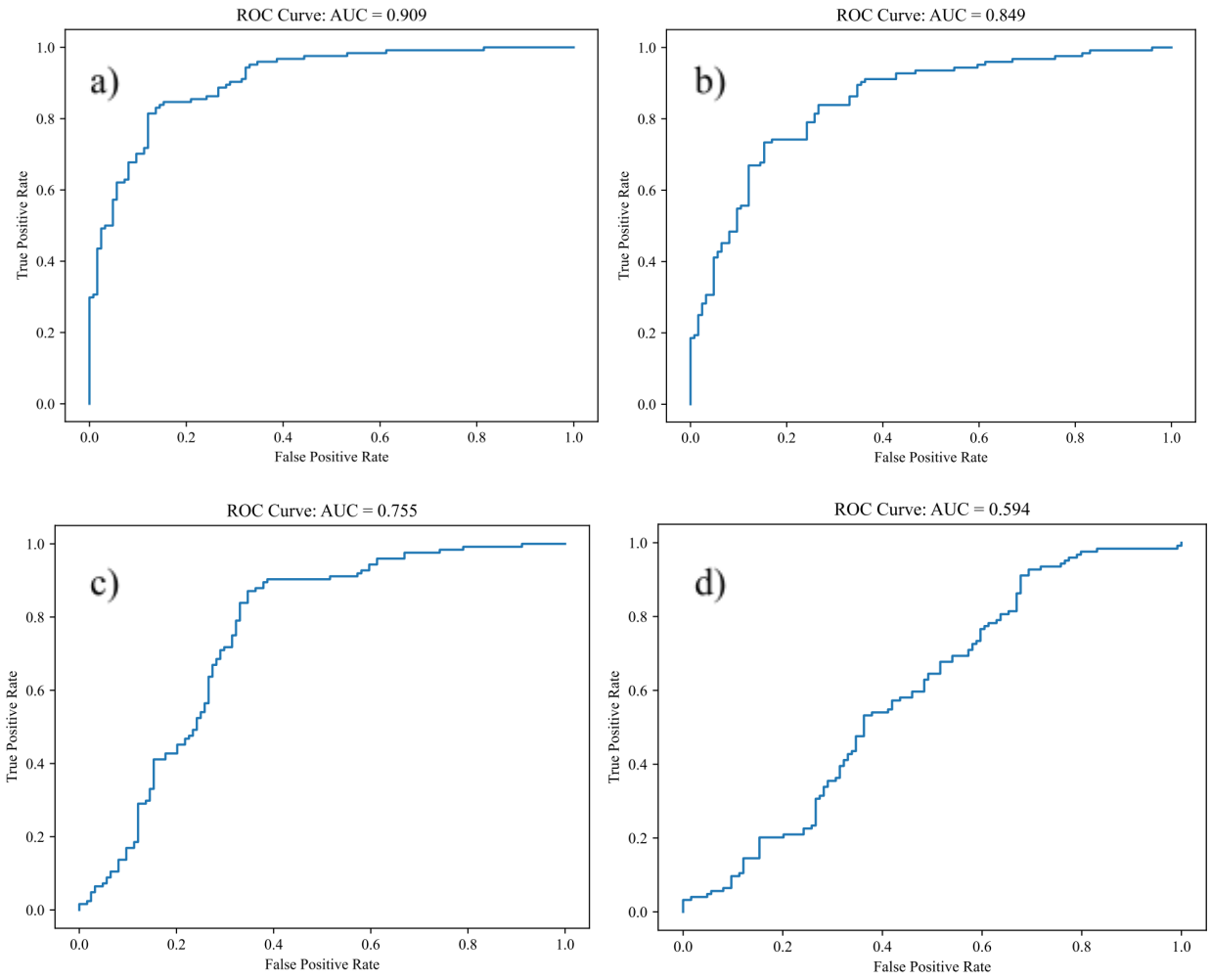
Model activations for real hotspot defects show the model's capacity to localize defects across several wafer technologies. While true, false negative cases still exist where activations seem to come from unseen chip technologies the model has not learned to bypass. With the integration of more training data, we expect the model's activations to become more precise across a wider array of technologies. Figures 10-17a and 10-17b show the training and validation accuracy of configuration 4's five folds as trained with a 0.001 weight decay.



Figures 10-17a and 10-17b: a) Configuration 4 training accuracy over 80 epochs, b) Configuration 4 validation accuracy over 80 epochs.

Training accuracy climbs asymptotically approaching one for all five folds across 80 epochs. Likewise, the validation accuracy follows, except plateauing around a score of 0.9. This model is trained across defect types, where it only sees synthetic data for the non-hotspot defects. Figures 10-18a through

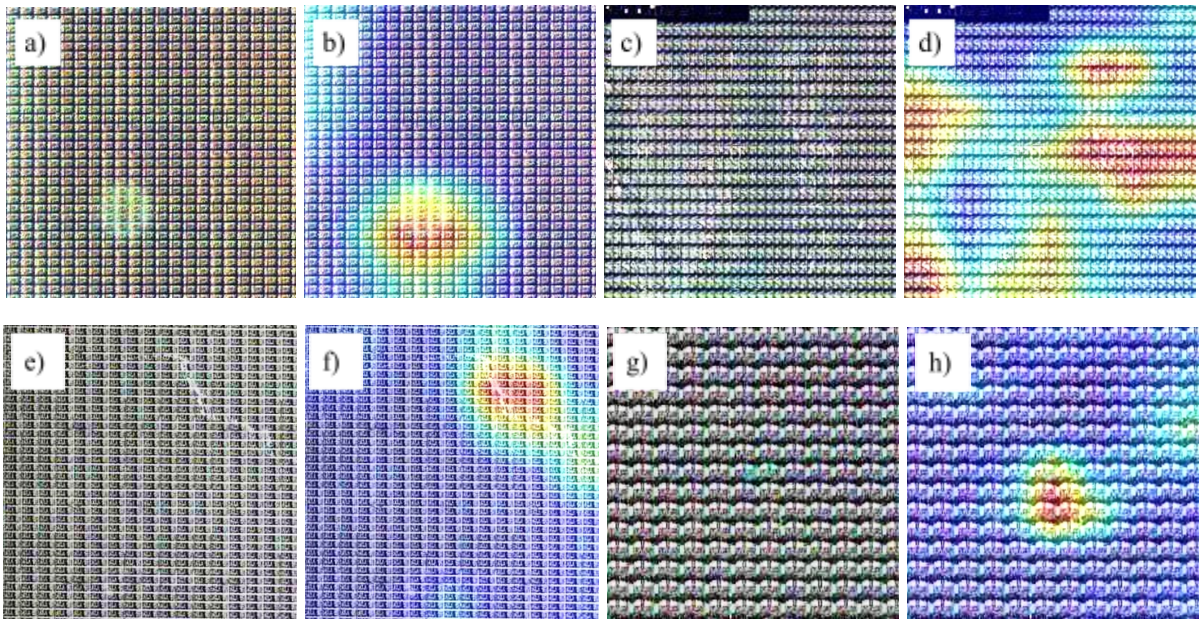
10-18d show the ROC curves for model performance on out-of-distribution data for each of the 4 defect classes targeted.



Figures 10-18a, 10-18b, 10-18c, 10-18d: a) Configuration 4 ROC curve for out-of-distribution hotspots, b) Configuration 4 ROC curve for out-of-distribution centerholes, c) Configuration 4 ROC curve for out-of-distribution contaminations, d) Configuration 4 ROC curve for out-of-distribution scratches.

The model's ability to identify hotspots improved noticeably between configurations 3 and 4 to an AUC of 0.909. This is likely due to the increased amount of data the model is trained on. The ROC curve for centerholes is also impressive, with an AUC of 0.849. This is surprising as the model was never exposed to a real centerhole. Being that some centerholes look like hotspots, this is a logical outcome though. Out-of-distribution testing for contaminations (AUC = 0.755) and scratches (AUC = .594) performed better than random guessing, but not as well as the hotspot models. In the future, as more data for these defect classes arises, it is expected that the AUC scores will rise across defect types as seen in hotspots. Additionally, as mo c) l defects are tagged, a multi-class model can be developed for the

identification of each defect type. Figures 10-19a through 10-19h show examples of true positives for each defect type in the out-of-distribution test sets, along with accompanying GradCAM activations.



Figures 10-19a, 10-19b, 10-19c, 10-19d, 10-19e, 10-19f, 10-19g, and 10-19h: a) Configuration 4 and out-of-distribution true positive hotspot example, b) GradCAM of 10-16a, c) Configuration 4 out-of-distribution true positive contamination example, d) GradCAM of 10-16c, e) Configuration 4 and out-of-distribution true positive scratch example, f) GradCAM of 10-16e, g) Configuration 4 out-of-distribution true positive centerhole example, h) GradCAM of 10-16g.

Across defect types, model activations for these true positives are highest where the defects are present, which is a promising result. While promising, these are just examples of the true positives. There are many cases like this for hotspots and centerhole defects, but for the worse performing scratches and contaminations, there are fewer instances of proper model activations. This highlights the need for more data showing real defects across technologies.

10.4 Limitations

Data is especially limited for center defects. As displayed, an increase in training data, whether synthetic or real, leads to an increase in out-of-distribution performance for each model. Due to the especially sparse nature of center defects, models couldn't be trained using any real defects other than hotspots, otherwise, there would be no data to withhold for out-of-distribution testing. Once OwlView is

implemented on the production floor, the amount of available and tagged data will grow rapidly, and model performance should follow as the data is retrained.

10.5 Chapter Summary

Data is especially limited for center defects. Regardless, respectable models were trained for detecting hotspot defects - with the best model achieving an out-of-distribution AUC of 0.909. Models showed layer activations corresponding to the location of each defect on the wafer for many instances. Better performing models had higher rates of true positives with proper activations. The major limitation of this model is the sparse amount of training data available.

Chapter 11

In-line Inspection Evaluation and Findings

In this section, only a high-level overview of the in-line inspection evaluation is covered. For a more detailed analysis, please refer to Cheung [35].

11.1 Summary of In-line Inspection Motivation and Methods

DFAB is in the process of converting to a 200mm only fab that produces primarily GaN technologies. To meet this end, management hopes to shift away from the flawed EAGLEView toolsets. While other out-of-line inspection tools could be integrated into the fab, a second and more compelling option exists. This is the integration of in-line inspection capabilities.

In-line inspection entails capturing and inspecting wafer images while on the toolsets that produce the defects. In the case of macro inspection, this occurs on toolsets that have coated the wafers with photoresist, and selectively cured the die geometry subsequently. By capturing images in-line with other processes, wafers do not need to be diverted to the EAGLEView module, saving both time and resources. Additionally, there is the potential to increase the overall quality of wafer images captured, and thus the inspection quality as well. To assess the feasibility of in-line macro inspection post photolithography at DFAB, a trial is conducted using a Keyence Line Scan camera and paired Lumitrax light bar as shown in Figure 11-1.

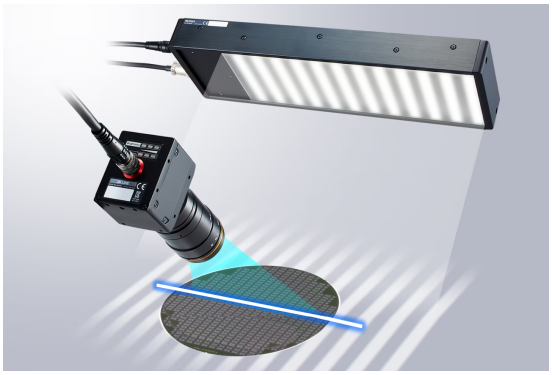


Figure 11-1: Keyence Line Scan and Lumitrax light bar [35]. The Line Scan camera captures images of wafers as the production tool moves them from place to place.

This system captures wafer images as they move along a single axis. The Lumitrax system shines different types of light on the wafer surface that are then used to create several different wafer images for the same wafer. The result is a high-quality image of the wafer with no reflection or glare, something that is near impossible to achieve without the Line Scan camera or Lumitrax. Additionally, the cameras come with rules-based detection algorithms that can be programmed by engineers for defect detection. Ideally, this can be used in parallel with ML algorithms trained on data collected upon implementation. As a result, specialists can still use OV, but rather than EAGLEView scans, scans come from the in-line tools.

This system is trialed on a TEL coat and development tool, as TELs are superior to other work cells in terms of cycle time, throughput, and downtime. Additionally, as DFAB shifts to a 200mm GaN fab, TELs will be utilized more often and more may be added to the fab. So, to evaluate the efficacy and feasibility of in-line inspection for macroscale defects, the mounting system shown in Figure 11-2 is designed and prototyped by our team.

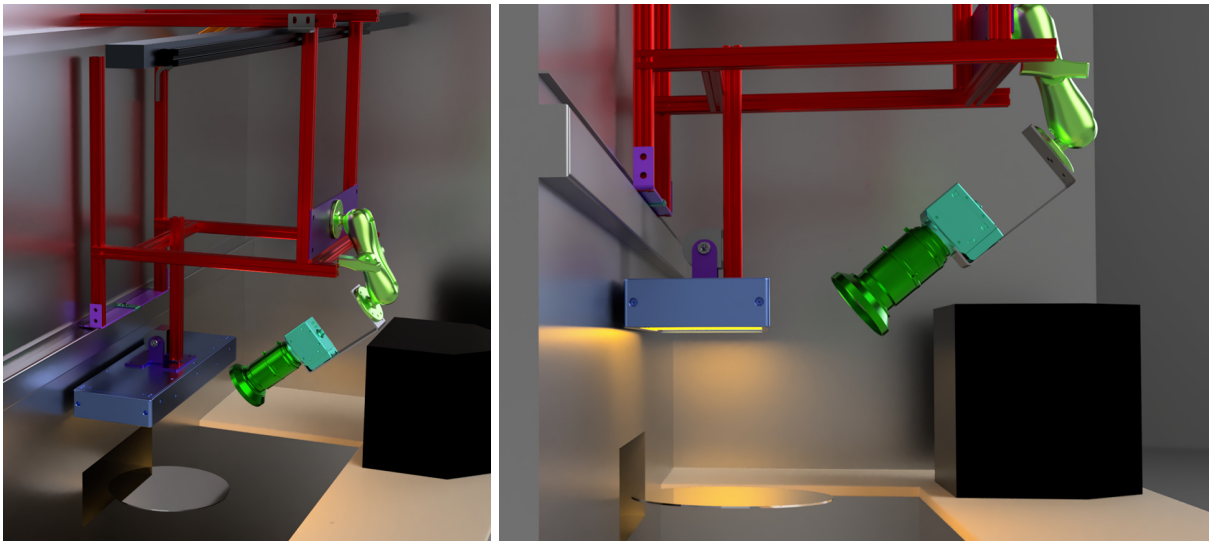


Figure 11-2: In-line inspection TEL mounts [35]: CAD rendering of inline inspection mount as installed in a tool.

The above mount is designed so that it may be integrated into any of the TELs present at DFAB. Additionally, the camera and lightbar positions are easily adjustable, a key requirement for the trial. Finally, an amber filter is added to the lightbar to filter out any light wavelengths that could lead to unwanted curing of the UV-sensitive photoresist. The described camera and lighting system is trialed for a one-week period. In this period, the image quality and system feasibility are both evaluated.

11.2 Summary of In-Line Inspection Trial

Figure 11-3 below shows images of the trial imaging unit as installed on the tool. The camera and light are controlled by, and write images to a computer next to the tool in the trial. The mount's modularity allows for adjustment of the camera and lighting angle, both of which are crucial for proper imaging of the wafer. Images are captured of the wafer as they are moved into the tool by a linear actuator. Example images captured during the trial are shown next in Figure 11-4.

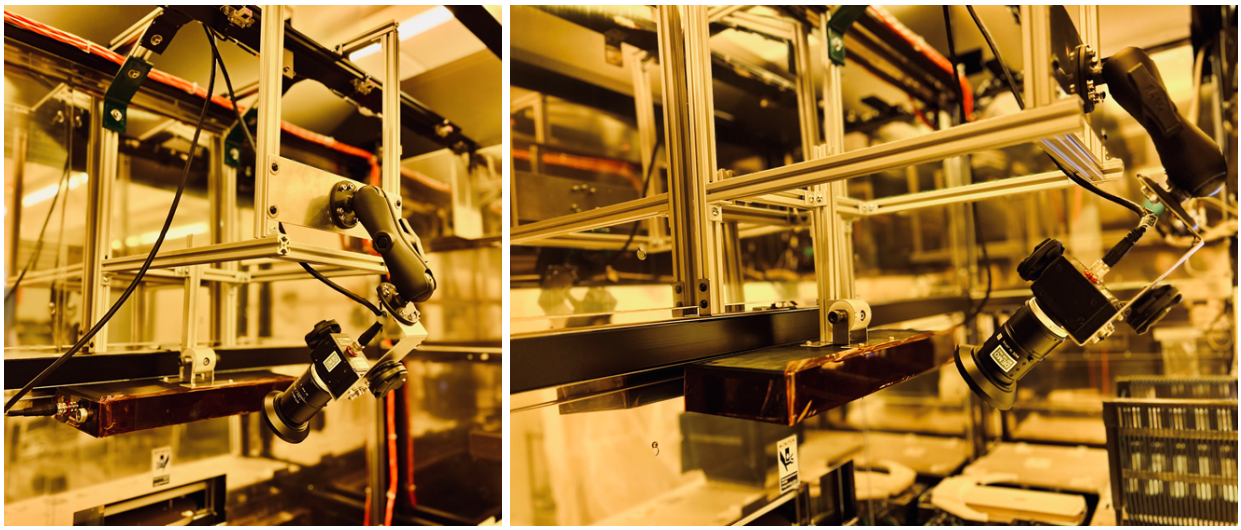


Figure 11-3: Installation of the mount on a TEL tool [35].

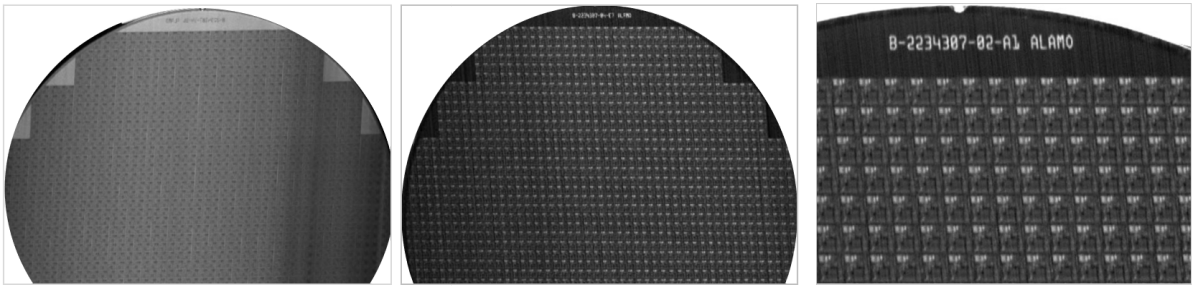


Figure 11-4: Wafer images collected during tool calibration of the in-line imaging system [35].

Images captured during the trial are partial given some limitations in the available lighting equipment. To achieve a full wafer image, the light would either need to be lowered past its current position - which is maximized in the current mounting arrangement - or be upgraded to a larger unit. Regardless, the image quality captured on the imaging system is promising, and certainly to par with the

EAGLEView system in resolution and quality. The trial unit also involved built-in defect detection algorithms, which are rules-based. Figure 11-5 below shows examples of the rules-based approach.

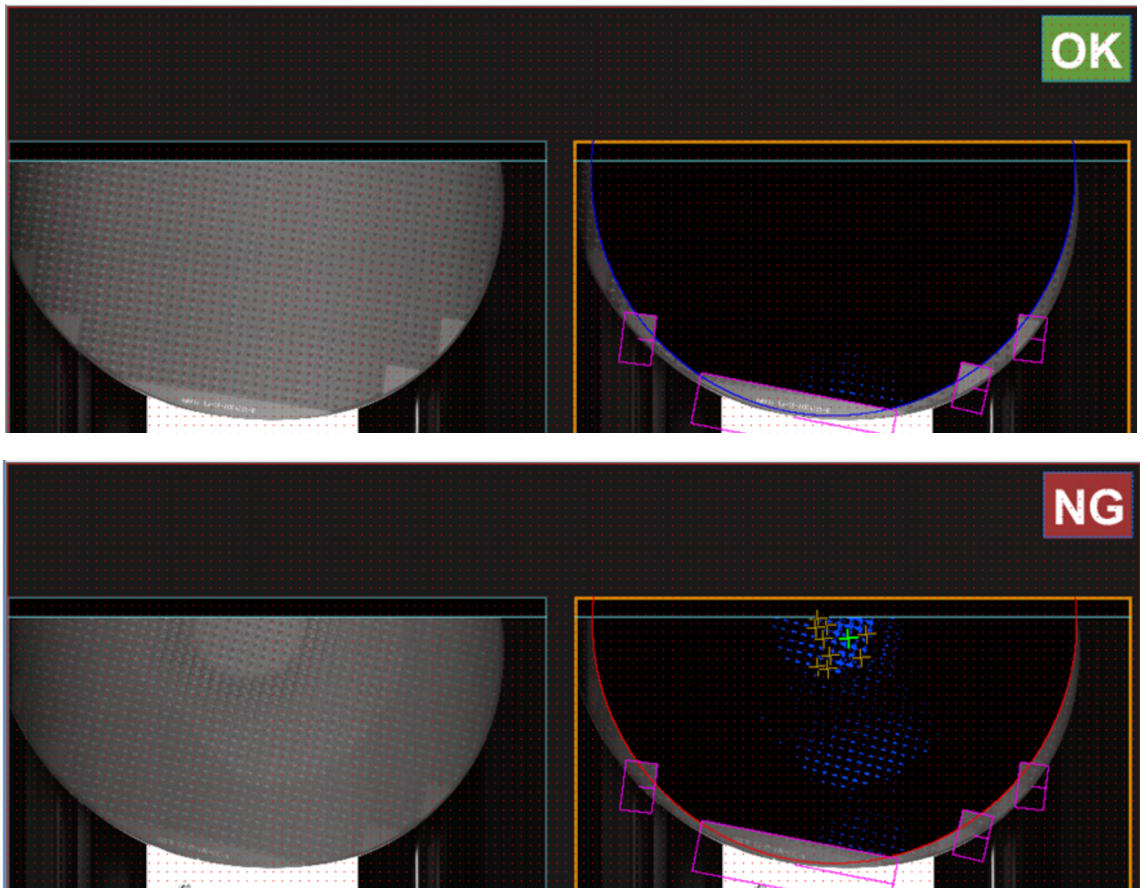


Figure 11-5: Keyence software detection images. No defect detected in the OK image (top), and a radial surface defect was detected in the no good image (bottom).

In this example, the Keyence system correctly identifies the presence of a spin defect in the bottom image and passes a good wafer in the top image. The pass or fail determination is made by user set rules. The rules set are based on radial variations in contrasts, and can be refined to identify different types of defects. The drawback of this approach is the need to set many rules for the many types of defects, but the trial proved feasibility. For a more thorough analysis, a larger dataset of defect types across wafers and layers is needed to create rules. Given EAGLEView tools utilize a rules-based approach, it is reasonable to assume proper rules could be devised to achieve production ready accuracy metrics.

11.3 Summary of Future Work

The trial proved the capacity to mount an imaging system that captures high-quality wafer bitmaps - which is no trivial task given the reflective nature of wafers. Additionally, this is done with no increase in wafer cycle time, which can help drive down the DPL of DFAB. Further, the rules-based approach to finding defects shows promise. More data on different defect types is required to set proper rules that catch defects. Work is also to be done in integrating the outputs of this in-line system to the new user interface OwlView. Ultimately, once many wafer images are collected and analyzed for defects, data can be used to train ML models with the same methodology described earlier in this thesis.

Chapter 12

Conclusions

Key results and conclusions of this thesis are first summarized, then areas for future work are discussed.

12.1 Summary of Results and Conclusions

Major improvements are made across four axes for DFAB's macro inspection module. The primary axis explored in the work is machine learning for defect detection. Binary classifier models are developed for SOG defects, edge defects, and central defects achieving out-of-distribution AUCs of 0.927, 0.906, and 0.909, respectively. Given the limited nature of defect data, particularly for center defects, transfer learning and synthetic data techniques are applied to improve model performances. GradCAM is used to visualize each model's ability to localize defect features, further supporting each model's ability to identify defective wafer regions. Results show a high potential for real-time ML-based defect detection at DFAB, but more data is required for better performance. This is a result of the wide array of different technologies, layers, and defect classes that exist. As a better representation of defects across technologies and layers is fed into the model, the out-of-distribution AUCs are expected to climb. In the development of these models, a machine learning infrastructure is created for future use by TI. This includes the new specialist interface, OwlView, for data tagging, our spatial tagger, sorting and splitting scripts, and more. Beyond machine learning, a new user interface called OwlView is developed based on specialist needs. Enhanced features like color equalization filtering, retroactive logpoint tracking, and more are added to make specialists more effective. Further, SOPs are updated to improve specialist performance. Steps are taken to implement OwlView on the production floor, and specialists are trained in the software. GR&R is conducted and various root causes of systematic shifts in EAGLEView performance are identified and mitigated. Procedures are put in place for recurring studies as well. Results from the in-line inspection trial show high quality image capture without the need for EV. Detection algorithms employed by Keyence also show promise in detecting spin defects - based on the small dataset of images the company utilized their rules-based approach on.

12.2 Future Work

Models developed in this work are trained on relatively limited data. Pending a PC upgrade in the fab, the new user interface, OwlView, and accompanying models are set to be deployed. Upon implementation, every decision stored by the operator can be utilized to sort data and multiply the amount of training data over time. By retraining after a set interval, model performances are expected to climb. Additionally, as the data set grows, new multiclass models can be trained for defect identification and classification. The metadata stored for each image can also be used in ML training in future models. Information including the tools that the wafer ran on before and the defect class can potentially be used as model inputs for increased accuracy and more insight to root cause tool failures, leading to systematic variations in product quality. As more GR&R studies on EAGLEView performance are conducted across layer levels and technologies, more sources of systematic tool performance are expected to be identified - leading to an increase in correct defect localization, and a decrease in false flagging rates.

PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Burkacky, Ondrej, and Nikolaus Lehmann. “The Semiconductor Decade: A Trillion-Dollar Industry,” McKinsey & Company, New York City, New York, 2022.

[2] Hufbauer, Gary Clyde, and Megan Hogan. “CHIPS Act will spur US production but not foreclose China.” *Peterson Institute for International Economics Policy Brief,* 22-13 (2022).

[3] Han, Jaecheon, et al., “Optimizing n-type contact design and chip size for high-performance indium gallium nitride/gallium nitride-based thin-film vertical light-emitting diode.” *Materials Science in Semiconductor Processing* 31 (2015): 153-159.

[4] Boles, Timothy. “GaN-on-silicon present challenges and future opportunities.” *12th European Microwave Integrated Circuits Conference (EuMIC)*. IEEE, 2017.

[5] Bordogna, A., et al., “Novel post-lithography macro inspection strategies for advanced legacy fab challenges.” *Metrology, Inspection, and Process Control for Microlithography XXXIV*. Vol. 11325. SPIE, 2020.

[6] Sampson, Jonathan A. *Improving Macroscale Defect Detection in Semiconductor Manufacturing using Automated Inspection with Convolutional Neural Networks.* Sep. 2023. Massachusetts Institute of Technology, Master’s of Engineering Thesis.

[7] Riverbank Computing Limited. “PyQT.” Version 5.15.9, Riverbank Computing Limited, 2012

[8] Bockstein, Ilia M. “Color equalization method and its application to color image processing.” *JOSA A* 3.5 (1986): 735-737.

[9] Huang, Szu-Hao, and Ying-Cheng Pan. “Automated visual inspection in the semiconductor industry: A survey.” *Computers in Industry* 66 (2015): 1-10.

[10] Lewis Jr, Richard E., Vinayan C. Menon, and Vandana Vishnu. “Implementation of integrated auto macro-defect inspection in the photolithography tool cluster.” *Metrology, Inspection, and Process Control for Microlithography XIX*. Vol. 5752. SPIE, 2005.

[11] Chang-Cheng Hung, Jhubei, and Hsinchu Tsai-Sheng Gau. *METHOD and SYSTEM for WAFER INSPECTION (U.S. Patent No. 8,617,410)*. 31 Dec. 2013.

[12] Lincoln, Schick. *Keyence Camera Trial Meeting*. 23 June 2023.

[13] Wu, Ming-Ju, Jyh-Shing R. Jang, and Jui-Long Chen. “Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets.” *IEEE Transactions on Semiconductor Manufacturing, Vol.* 28, No. 1 (February 2015): 1–12.

[14] Gong, Jie, and Chen Lin. “Wafer Map Failure Pattern Classification Using Deep Learning."

[15] Chen, Yu, et al., “Wafer defect recognition method based on multi-scale feature fusion.” *Frontiers in Neuroscience* 17 (2023): 1202985.

[16] Shinde, Prashant P., Priyadarshini P. Pai, and Shashishekar P. Adiga. “Wafer defect localization and classification using deep learning techniques.” *IEEE Access* 10 (2022): 39969-39974.

[17] Chen, Yu, et al., “An Auto-adjusting Weight Model for Imbalanced Wafer Defects Recognition.” *2022 International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD)*. IEEE, 2022.

[18] Alawieh, Mohamed Baker, Duane Boning, and David Z. Pan. "Wafer map defect patterns classification using deep selective learning.” *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020.

[19] Liukkonen, Mika, and Yrjö Hiltunen. “Recognition of systematic spatial patterns in silicon wafers based on SOM and K-means.” *IFAC-PapersOnLine* 51.2 (2018): 439-444.

[20] Facebook AI Research. “PyTorch.” Version 3.8.1, Facebook AI Research, 2023

[21] Simonyan, Karen, and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition.” *arXiv preprint arXiv:1409.1556* (2014).

[22] PyTorch. “VGG11.” Version 3.8.1, Facebook AI Research, 2023

[23] Yosinski, Jason, et al., “How transferable are features in deep neural networks?.” *Advances in Neural Information Processing Systems* 27 (2014).

[24] ImageNet Large Scale Visual Recognition Challenge (ILSVRC). “ILSVRC data set.” 2012-2017, ImageNet, http://image-net.org/challenges/LSVRC/

[25] Kingma, Diederik P., and Jimmy Ba. “Adam: A method for stochastic optimization.” *arXiv preprint arXiv:1412.6980* (2014).

[26] Dogo, Eustace M., et al., "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks." *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*. IEEE, 2018.

[27] Srivastava, Nitish, et al., "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research* 15.1 (2014): 1929-1958.

[28] Bergmeir, Christoph, and José M. Benítez. "On the use of cross-validation for time series predictor evaluation." *Information Sciences* 191 (2012): 192-213.

[29] Kumar, Ananya, et al., "Fine-tuning can distort pre-trained features and underperform out-of-distribution." *arXiv preprint arXiv:2202.10054* (2022).

[30] Wong, Sebastien C., et al., "Understanding data augmentation for classification: when to warp?." *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2016.

[31] TorchVision. "TorchVision: Transforms and Augmentation Functions." Version 0.15, PyTorch, 2023.

[32] Seib, Viktor, Benjamin Lange, and Stefan Wirtz. "Mixing Real and Synthetic Data to Enhance Neural Network Training--A Review of Current Approaches." *arXiv preprint arXiv:2007.08781* (2020).

[33] Tan, Aik Jun. *Deep Learning Image Augmentation using Inpainting with Partial Convolution and GANs*. Masters Thesis, Massachusetts Institute of Technology, 2021.

[34] Selvaraju, Ramprasaath R., et al., "Grad-cam: Visual explanations from deep networks via gradient-based localization." *Proceedings of the IEEE International Conference on Computer Vision*. 2017.

[35] Cheung, Sophia. *Machine Learning Methods for Automated Macro-Inspection and Improved Defect Identification in Semiconductor Manufacturing.* Sep. 2023. Massachusetts Institute of Technology, Master's of Engineering Thesis.