WAFER MAP DEFECT PATTERN CLASSIFICATION USING DEEP LEARNING MODEL

BY

LIM YU PIN

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2023

WAFER MAP DEFECT PATTERN CLASSIFICATION USING DEEP LEARNING MODEL

BY
LIM YU PIN

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfilment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2023

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: Wafer Map Defect Pattern Classification using Deep Learning Model

Academic Session: MAY 2023

I LIM YU PIN

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:
142, Jalan Binjai
Taman Sri Rambai
14000 Bukit Mertajam
Pulau Pinang

Lim Jia Qi
Supervisor's name

Date: 12/09/2023

Date: 14/09/2023

Universiti Tunku Abdul Rahman			
Form Title: **Sample of Submission Sheet for FYP/Dissertation/Thesis**			
Form Number: **FM-IAD-004**	Rev No.: **0**	Effective Date: **21 JUNE 2011**	Page No.: **1 of 1**

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

UNIVERSITI TUNKU ABDUL RAHMAN

Date: 12/09/2023

SUBMISSION OF FINAL YEAR PROJECT

It is hereby certified that Lim Yu Pin (ID No: 19ACB06069) has completed this final year project entitled "*Wafer Map Defect Pattern Classification using Deep Learning Model*" under the supervision of Dr. Lim Jia Qi (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,

____

(Lim Yu Pin)

DECLARATION OF ORIGINALITY

I declare that this report entitled “**WAFER MAP DEFECT PATTERN CLASSIFICATION USING DEEP LEARNING MODEL**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature: ____________________

Name: Lim Yu Pin

Date: 12/09/2023

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor Dr Lim Jia Qi for their guidance, support, and encouragement throughout the duration of my final year project. Their expertise and insights were invaluable in shaping the direction and focus of my research. I am truly fortunate to have had such a dedicated and knowledgeable mentor.

I would also like to extend my heartfelt appreciation to my parents and family members for their unwavering love, encouragement, and support. Their constant encouragement and belief in me were the driving force behind my success in completing this project.

Finally, I would like to express my gratitude to all the giants upon whose shoulders I have stood, as their research have provided invaluable insights and knowledge for my research. Their contributions in the field were significant and greatly appreciated.

Thank you all for being a part of my journey and for making it such a memorable and rewarding experience.

ABSTRACT

Wafer maps are generated during wafer testing in the semiconductor manufacturing process. They contain valuable information that helps engineers identify faults in the fabrication process. Classification of defect patterns is necessary to identify the root cause of die failures, and deep learning models have shown promising results in this regard. However, traditional CNN models have limited ability to handle the varied distribution of defect patterns in different wafer maps. The absence of balanced wafer map defect patterns dataset also posed a challenge to the training of CNNs.

In this research, a novel approach that combines Connected-Component Labelling (CCL) for noise reduction, Convolutional Autoencoders (CAE) for data augmentation to address dataset class imbalance issue, and transfer learning via the EfficientNet model for an end-to-end system capable of accurately classifying wafer map defect patterns has been proposed. Experimental results showed that the proposed model demonstrates robust performance in terms of accuracy, precision, recall and F1-Score, which confirmed its effectiveness in classifying wafer map defect patterns.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

API	Application Programming Interface
ART1	Adaptive Resonance Theory Network 1
CCL	Connected-Component Labelling
CNN	Convolutional Neural Network
GAP	Global Average Pooling
GMP	Global Max Pooling
GPU	Graphics Processing Unit
IC	Integrated Circuit
IDE	Integrated Development Environment
NR-A-GAP	Noise Reduction, Augmented, Global Average Pooling
NR-A-GMP	Noise Reduction, Augmented, Global Max Pooling
NR-NA	Noise Reduction, Non-augmented
ReLU	Rectified Linear Unit
RGB	Red Green Blue
RU	Raw, Unaltered
SOM	Self-Organising Map
SVM	Support Vector Machine

CHAPTER 1 INTRODUCTION

1.1 Background Information

Before die preparation and integrated circuit (IC) packaging in the semiconductor device fabrication process, wafer with fabricated die must go through wafer testing where defects on each die can be detected. These defects can be caused by process variation which are variations in the length, widths and oxide thickness of transistors that affects the performance of the circuits [1]. As semiconductor technology continues to scale according to Moore's Law, the impact of process variation is even more profound as technology node goes beyond the sub-90 nm process [2]. Therefore, wafer testing is a key step in the fabrication process to detect defect patterns which can limit product yield that is the percentage of working dies on any wafer. Furthermore, the main goal of the proposed transition towards 450 mm wafers from the current 300 mm wafers is cost reduction. Wafer testing is therefore a vital step in reducing the number of discarded wafers by identifying the root causes of the defects.

Wafer map, which is a simple graphical representation of chips on wafer can provide key information on the locations of defective dies on a silicon wafer [3]. During wafer testing, each die on the wafer is electrically tested by wafer probe and assigned a bin value. The bin value is assigned based on whether the individual die passes or fails the test. If the individual die passes, it is then sent to be diced and packaged during later fabrication stages. The spatial pattern of the cluster can provide valuable information to determine the root cause of defects. Thus, analysis of defect spatial patterns in wafer map is useful for monitoring the quality of manufacturing process.

In view of the above arguments regarding the significance of wafer map in detecting the root cause of defects on the silicon wafer, deep learning model which can learn robust feature representation is proposed to classify the defect patterns of wafer map.

1.2 Problem Statement

In traditional semiconductor device fabrication process, wafer surface is manually reviewed by human experts during visual inspection process to detect defects [4]. The process is tedious and slow, often requiring concentration and skills by the inspector. With the increased production rate of up to thousands of wafers per week in modern fabs, it is impractical for inspectors to manually inspect each wafer map to categorise them into different defect patterns. The proposed transition to 450 mm wafers would make visual inspection process even more obsolete as larger wafer would be able to generate more dies per wafer [5]. The traditional approach is therefore not feasible as it requires investment into training inspector for defect classification task and there is a high probability of variation and bias in classification even when inspection is conducted by the same inspector.

Numerous techniques have been proposed for defect patterns metrology. Some early methods employ the use of statistical analysis to detect defect patterns [6], [7]. However, these methods were proposed for the purpose of detecting the occurrence of wafer defect patterns without further classifying them into different categories of patterns. Since then, with the advent of machine learning, a wide range of techniques has been used for the purpose of classifying defect patterns.

Machine learning algorithms, including support vector machine [8], [9] and *K*th nearest neighbour [10] have proven effective in the recognition of defect patterns. However, these methods rely heavily upon small-scale, quality data with low amount of noise and therefore require significant feature engineering. Wafer map datasets that are available in public domain vary in quality and there exists no predefined definition on how normal and defective dies are mapped as they were acquired with different image acquisition methods. Extensive data pre-processing such as noise filtering and feature selection must be done beforehand and would result in the loss of information causing reduced accuracy.

The recent development of deep learning has prompted the application of the technique on tasks that requires learning complex raw data. The ability to learn features end-to-end by training with raw data makes deep learning method a significant improvement from traditional machine learning that depends on manual feature engineering. Convolutional neural networks (CNN) have been used extensively and is the current state of the art method for image recognition and classification. While CNNs shows promising and reliable results as evident in the large amount of research focusing on its use for defect patterns classification, designing and training CNN for each specific task requires time, substantial computational resources, and large training dataset [11]. Furthermore, the scarcity of labelled wafer map datasets presents a problem to CNN's supervised learning paradigm.

As mentioned, we can see that most machine learning methods are dependent upon both training and test data that are taken from the same feature space and distribution. A complete rebuilding of models with newly acquired training data are often required when the distribution is changed. Furthermore, wafer map distribution varies in each dataset, defect patterns found in one dataset may not exist in another due to the complex nature of the semiconductor manufacturing process making the need to rebuild models each time a new wafer map dataset with different distribution is presented expensive and not feasible. There is also a lack of labelled data in wafer map datasets as domain experts are required to properly label these datasets.

Thus, accurate, reliable, and robust deep learning model which can perform on-line wafer classification and ultimately streamline the quality control process is much sought after. The use of transfer learning method can be used to mitigate the problems where the cost of redesigning and training a new CNN is high. A systematic review by [11] found that in the limited studies based on transfer learning approach have shown promising results in classifying defect patterns and should be investigated further. Therefore, a transfer learning approach based on pre-trained EfficientNet is proposed to perform on-line wafer classification and ultimately streamline the quality control process. EfficientNet is chosen due to its ability to scale its architecture in principled manner in terms of depth, width, and resolution which result in better performance while at the same time reducing the parameters of the network compared to other CNN architectures.

1.3 Project Scope

The aim of this project is to build a deep learning model based on transfer learning method using pre-trained EfficientNet that could receive wafer map images as input and to classify to classes the various defect patterns found in the wafer map.

1.4 Project Objectives

The objectives of this research are:

i. To investigate state-of-the-art deep learning models for wafer map defect patterns classification and their corresponding performance.
ii. To address the class imbalance problem found in wafer map dataset by using convolutional autoencoder to generate synthetic data.
iii. To mitigate the need to redesign and retrain models when new wafer map data are presented through transfer learning method.
iv. To build a wafer map classification system that can distinguish between different defect patterns based on features extracted from pre-trained EfficientNet and classification using custom layers.

1.5 Impact, Significance and Contribution

There are three main contributions from this project: (a) Extract robust and reproducible features for wafer map defect patterns classification using transfer learning method, (b) Alleviate the dependence of model on large amount of data required for wafer map defect patterns classification, (c) presents a robust wafer map defect patterns classification system. Considering that transfer learning method had not been explored extensively for wafer map defect patterns classification, this project plays an important role in propagating the use of transfer learning method in the field and acts as a benchmark for future research work.

1.6 Report Organisation

Chapter 1 serves as a detailed introduction to the project. It covers various aspects such as the problem statement, project scope, project objectives, and the impact, significance, and contribution of the project. Additionally, the background information provided in this chapter sheds further light on the project.

Chapter 2 comprises a literature review of previous studies that focus on the identification and classification of wafer map defect patterns. These studies include early unsupervised neural-network architecture, custom-made CNN, and transfer learning.

In Chapter 3, the proposed method is explained in detail, including the methodologies and system design. To aid in the understanding of the proposed system, a flowchart of the system design is provided.

Chapter 4 details the experimental results and analysis of the research, including data exploration, data pre-processing, model training and hyperparameter tuning. A thorough analysis of the results obtained from this stage is also presented.

Finally, Chapter 5 provides a conclusion on the project's findings and recommendations for the future research.

CHAPTER 2 LITERATURE REVIEW

2.1 Convolutional Neural Network

Convolutional neural network (CNN) is a class of artificial neural network that is inspired by the connectivity pattern between neurons in the animal visual cortex. It is primarily used for computer vision task such as image classification and object recognition. A convolutional neural network (CNN) is composed of layers that perform convolutions, activations, pooling, and fully connected operations, and is designed to extract spatial features from image data and classify them into different categories. There are three main types of hidden layers, which are convolutional layer, pooling layer, and fully connected layer.

An in-depth explanation of CNN hidden layers is provided in [12]. A convolutional layer takes the input data where in the case of a colour image consists of height, width and depth corresponding to the RGB colour space and pass them through a 3-dimensional filter that move across the image which then calculate the dot product between the input pixels and the filter to produce a feature map. Stride determines the distance at which the filter move over the image and the size of the output can be decreased by increasing the stride value. Zero-padding is used to prevent the loss of information that is outside of the image border to ensure that the output matches the size of the input. By setting the matrix element that fall outside of the input matrix to zero, the size of the input matrix can be preserved in the output matrix. In addition, non-linearity is used to limit the output by a decision boundary. Common non-linearity functions include sigmoid, tanh and Rectified Linear Unit (ReLU).

Following convolution, pooling is performed to down-sample the output from the convolutional layer to reduce the complexity for subsequent layers by averaging the matrix into a smaller size. After the convolutional layer and pooling layer, the matrix is flattened and passed to a fully connected layer where each node is connected directly to a node in the previous layer. A SoftMax layer is then added for class probability calculation to classify the output from the fully connected layer into specific classes. To visualise the layers in a CNN, Figure 2.1 shows the architecture for LeNet-5, often considered the pioneer in the field of CNN.

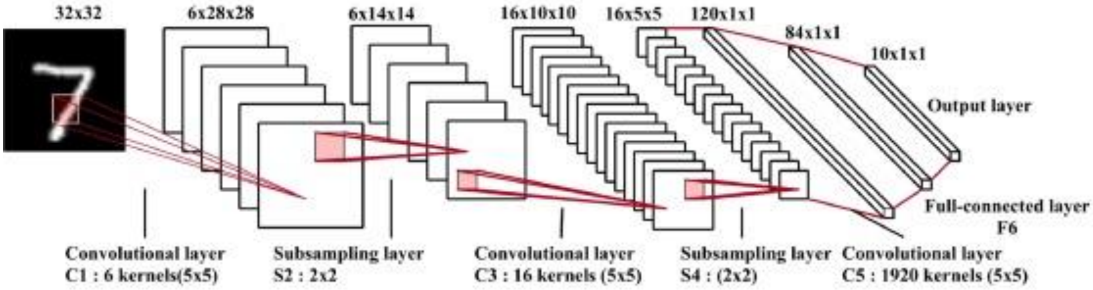


Figure 2.1 Architecture of LeNet-5.

As we can see, CNNs can be made up of repeating combination of convolutional layers, activation function and pooling layers. As the image passes through the multiple layers, the repeating structure can detect higher-level features than the previous layer from edges to complex shapes. Bottom layers often learn to identify simple, low-level features such as edges and corners, and as the input progresses to deeper layers, it can recognise more complex and high-level features such as shapes and objects. By learning a hierarchy of features, the underlying structure and patterns in the data can be captured by CNNs to enable the network to generalise better and be more robust to variations in the input data. Therefore, considerable focus on deeper CNN with more feature learning layers can be found in the literature, ranging from a few to hundreds of layers.

2.2 Transfer Learning and Pre-trained Models

Transfer learning refers to a machine learning technique where knowledge from previous tasks is transferred to a target task, often in cases where the target task has fewer training data to be used. This is different from traditional machine learning where a model is trained from scratch using existing data.

Pan and Yang [13] provided a definition of transfer learning:

> "Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $fT(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$."

Transfer learning initially involves pretraining models on a source data to encode its knowledge, which can subsequently be transferred to train models intended for a different tasks. In the convolutional neural network field, various models such as DenseNet [14], ResNet [15], and EfficientNet [16] that is pretrained with popular datasets such as ImageNet [17] is available.

2.3 Related Work

Before CNN is heavily used for defect spatial pattern classification in wafer map, one of the earliest works on using deep learning architecture for this task was introduced by Chen and Liu [18]. The approach taken uses unsupervised neural-network architecture called adaptive resonance theory network 1 (ART1). The reason behind the use of unsupervised learning in this research was that training templates were hard to come by and for its ability to solve the stability-plasticity dilemma. The authors highlighted several previous methods that either requires adequate representative training samples or was extremely time consuming. The use of vigilance test in ART1 will deal with these limitations. The vigilance test passes when the new pattern is found to be similar to previously stored pattern which will then be slightly modified and retained whereas the vigilance test fails when the new pattern is found to be not similar to any previously stored pattern then a new knowledge for the pattern will be created. Once the vigilance test is conducted and the network is trained, maximum match value was assigned to every new pattern. These match values were then compared with a pre-determined threshold, P_m, to classify them into specific cluster. Each possible situations for the comparison and their subsequent actions and results are shown in Figure 2.2.

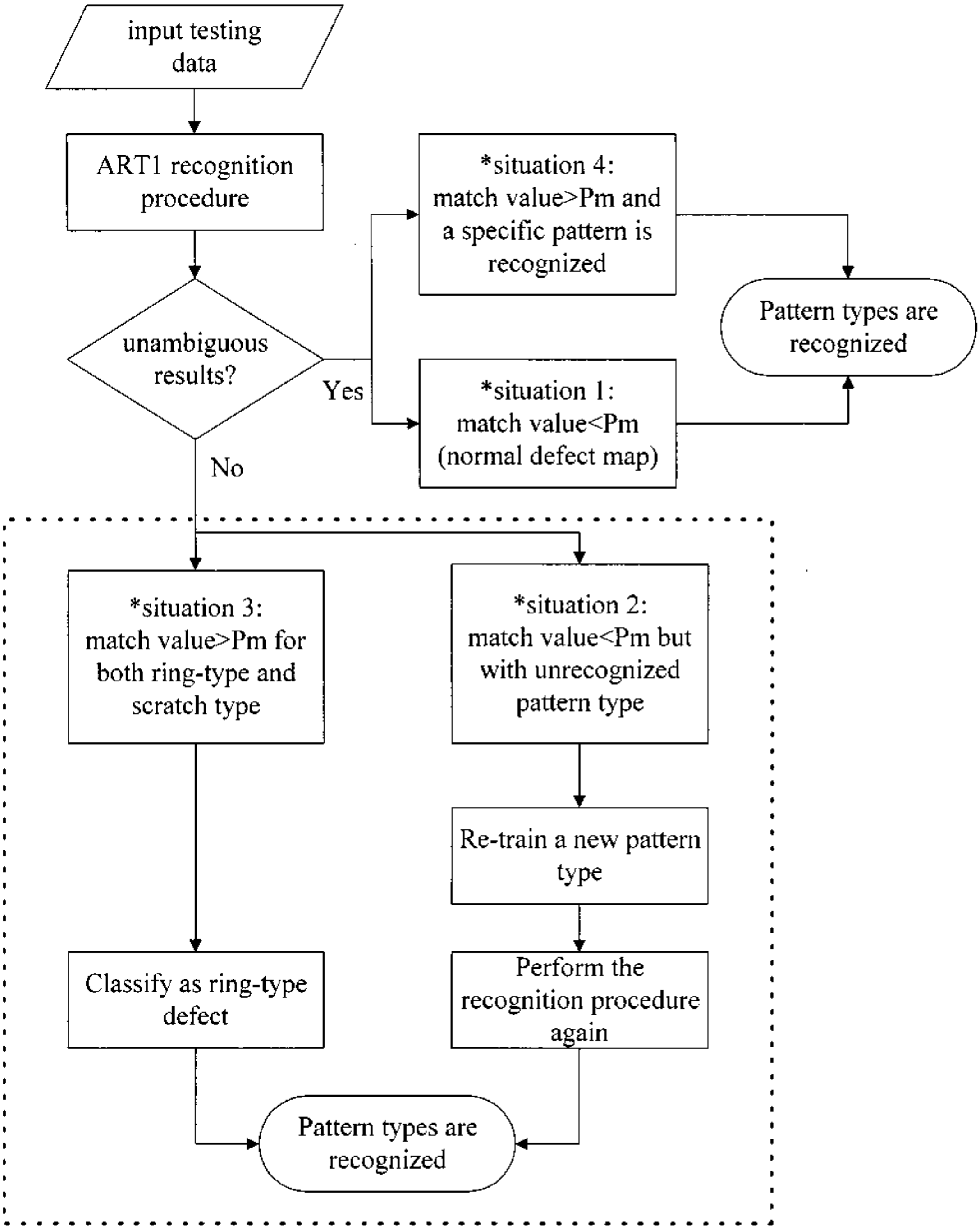


Figure 2.2 ART1 analysis possible scenarios.

When compared with the Kohonen self-organising map (SOM), another unsupervised network trained with the same dataset, the ART1 algorithm can converge in just five iterations compared to SOM which takes more iterations to complete. When applied on test data with 35 samples, ART1 can learn at a significantly shorter time at 3 seconds compared to 30 second for SOM. Another advantage that ART1 has over SOM is the ability to distinguish similar but different defect spatial patterns, such as ring type with different radii.

However, there are several limitations in this approach. The model is trained on only two types of defect spatial patterns of ring and scratch types. This is not an adequate representation of the diverse defect spatial patterns found during fabrication process and is impractical to be used in real world defect monitoring systems. Furthermore, the model will need to be retrained when more defect patterns are introduced. This contributes to more unnecessary time and resources spent on redesigning and retraining the model instead of focusing on the classification task.

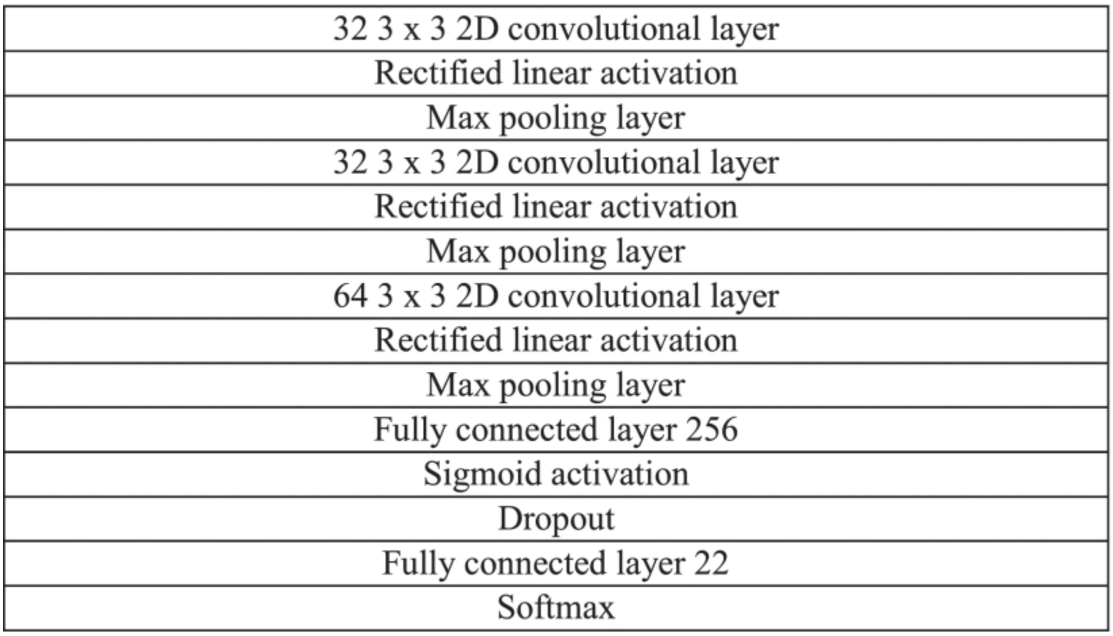
Since real data for each class of defect patterns are hard to come by and highly imbalanced, [19] presents an approach of generating synthetic wafer maps to be used for CNN training. The authors also visualised the wafer map based on the frequency density to provide more information regarding the occurrence rate of defect patterns.

Poisson point process is used to model the defect pattern and the distribution is given as

$$\mathrm{P}(\mathrm{k},\Lambda)=\frac{\Lambda^{\mathrm{n}}}{\mathrm{k}!}\mathrm{e}^{-\Lambda},$$

where Λ is the rate parameter and k is the number of events to generate the random points. This is followed by the superimposition of non-random points through control of the interval of uniform distribution. In total, 22 classes each representing different defect patterns ranging from random defects to non-random defects were generated.

The CNN configuration is shown in Figure 2.3. The CNN has 3 convolutional layers each that uses ReLU activation with 3×3 receptive field size, stride 1, and 2×2 max pooling size. After the convolutional layers, 2 fully connect layers was used in the model. The first fully connected layer is of size 256 with sigmoid activation function added. Dropout was then used for regularisation to reduce overfitting before being passed to the last fully connected layer with the size of the defect class. Decimal probabilities were then generated using SoftMax to be output.



32 3 x 3 2D convolutional layer
Rectified linear activation
Max pooling layer
32 3 x 3 2D convolutional layer
Rectified linear activation
Max pooling layer
64 3 x 3 2D convolutional layer
Rectified linear activation
Max pooling layer
Fully connected layer 256
Sigmoid activation
Dropout
Fully connected layer 22
Softmax

Figure 2.3 Configuration of CNN used.

The CNN achieved state of the art accuracy at 99.8% during training and 98.2% during testing. However, these numbers were obtained from simulated wafer map. Citing confidentiality reason, the authors was not able to provide overall accuracy when tested with real wafers data. Nevertheless, accuracy for the 9 classes found in the real wafer dataset shows that the model was able to classify most classes except for two rare defect classes.

While simulated wafer maps can serve as a substitute for real wafer map datasets and to provide for more variation to aid models, additional resources are required be set aside to generate them. Along with the need to redesign and train new CNN as mentioned previously, the approach taken by this paper is unsuitable to be deployed in fast-paced and complex semiconductor manufacturing process.

To reduce training time and address class imbalance problem, [20] presented a transfer learning based DenseNet model for wafer map defect pattern classification task. As with any feedforward neural network, CNN suffers from the vanishing gradients problem due to the implementation of backpropagation algorithm where the error gradient is propagated from the output layer to the input layer. As the model gets deeper, gradients get smaller until eventually the connection weights remain unchanged. As shown in Figure 2.4, DenseNet directly connects any layer to all subsequent layer and concatenate the feature maps produced by all preceding layers [14]. The ability to alleviate this problem is the primary reason that DenseNet is used.

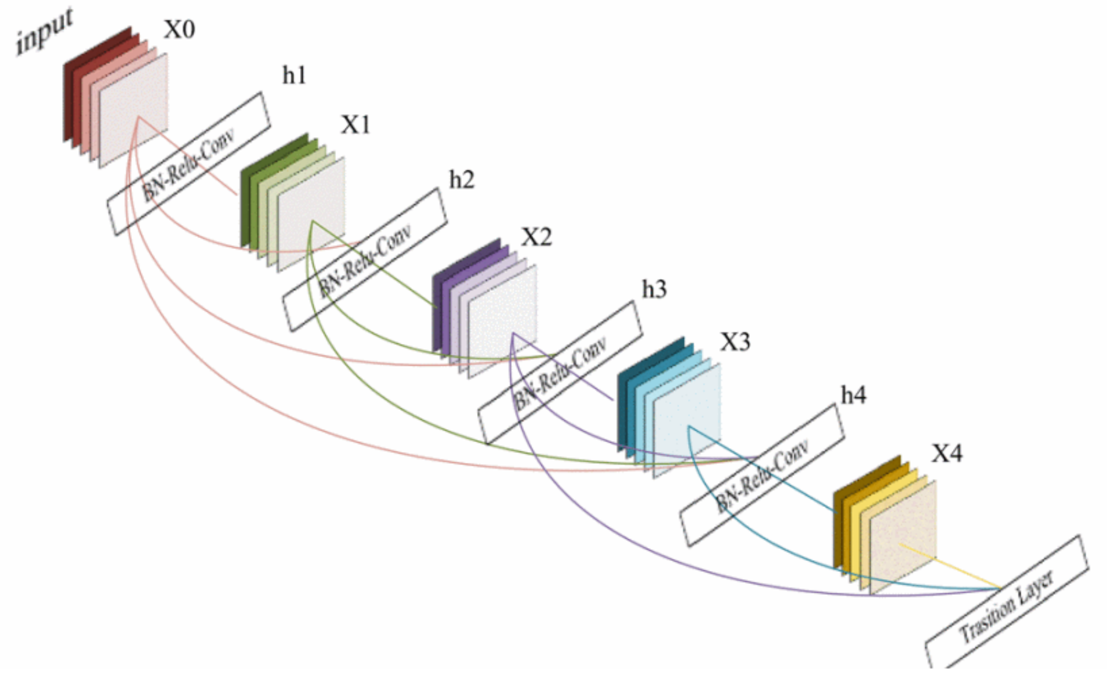


Figure 2.4 Architecture of DenseNet.

The research performed modelling by transferring the weight parameters of the large ImageNet dataset to DenseNet169, a variant of DenseNet with a depth of 169 and redesigning the classification layer to fit the classification task. Instead of an end-to-end model, the research used DenseNet as a feature extractor and employed the use of multi-grained cascade forest (GCForest) for classification. The reason for the use of GCForest is due its ability to improve learning from high dimensional feature that was extracted by DenseNet169 through multi-dimensional scanning and cascade processing method. The application procedure of the proposed DenseNet-GCF model is shown in Figure 2.5.

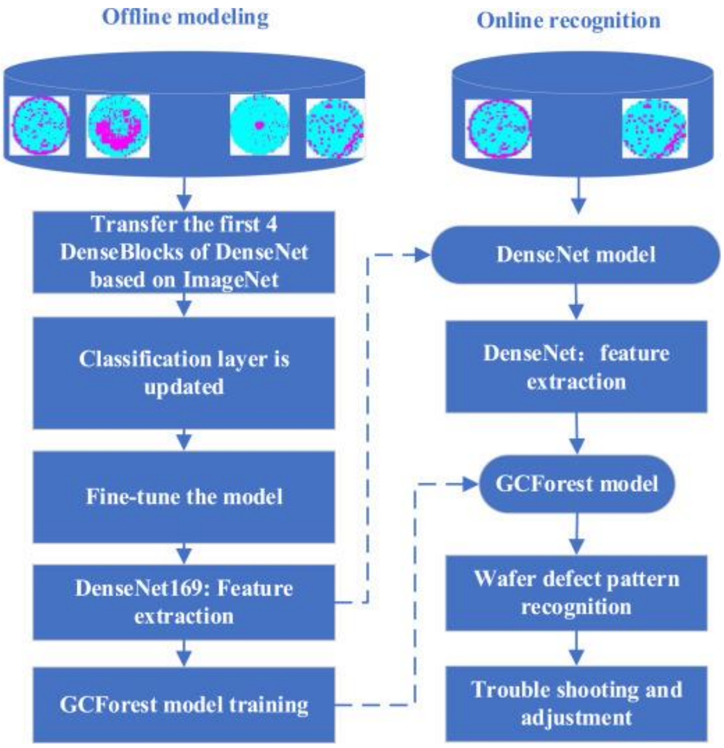


Figure 2.5 DenseNet-GCF application procedure.

The performance of the DenseNet model with and without transfer learning was evaluated and it is found that the transfer learning based DenseNet model performed better. The transfer learning model can achieve convergence earlier than the one without transfer learning and ultimately other CNN models. When compared with other classifiers, DenseNet-GCF were able to achieve a five-cross validation of 96.8% with ResNet, that achieved the second highest score falling behind significantly at 86.5%.

However, by excessively connecting the layers of an already deep model, DenseNet can decrease computation-efficiency and parameter-efficiency, while at the same time is more likely to overfit [21]. This is due to the increasingly replicated data when DenseNet progresses through its layers which can be resource intensive in terms of computation and memory.

2.4 EfficientNet

To overcome the limitations mentioned above, a new CNN model called EfficientNet can be used. The model is described in the paper "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" [16]. EfficientNet achieved better performance compared to previous models by uniformly scaling all dimensions of depth, width, and resolution instead of scaling these factors arbitrary as seen in previous models. It is found that through balancing these three dimensions by scaling them with a constant ratio, this method, called compound scaling can improve the efficiency of deep CNN. Figure 2.6 provides an illustration of how EfficientNet differs with other models in scaling the three dimensions.

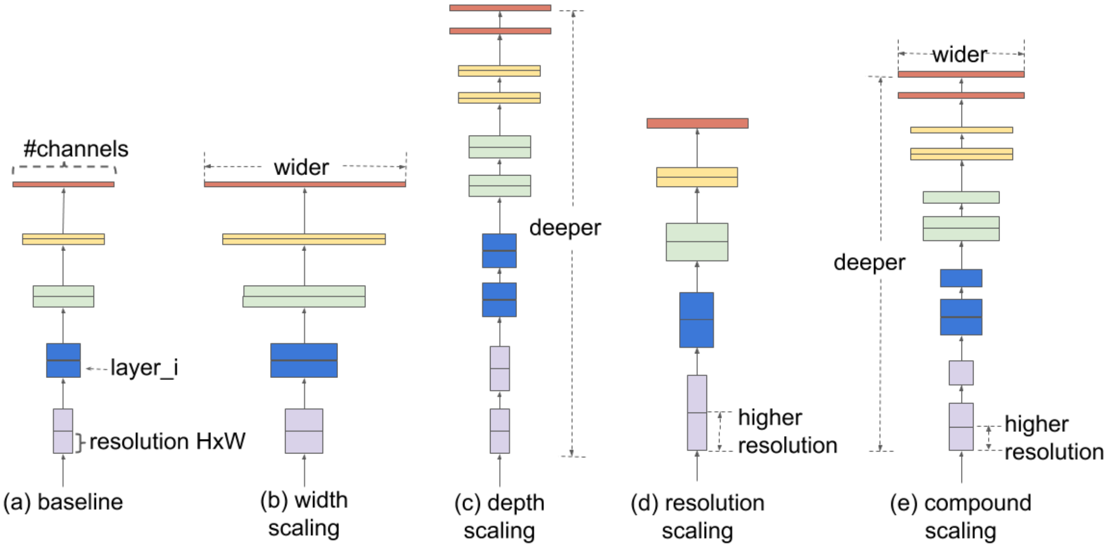


Figure 2.6 (a) shows the baseline network; (b)-(d) shows the conventional scaling that usually increases only one dimension of width, depth, or resolution; (e) shows the proposed compound scaling of EfficientNet.

The reason to balance all dimensions during scaling stemmed from the observation that when any one dimension is scaled up, the accuracy gain saturates in bigger models. From the baseline model EfficientNet-B0, the authors proposed a novel compound scaling method by scaling up the compound coefficient uniformly for all three dimensions to obtain EfficientNet-B1 to B7. The architecture for EfficientNet-B0 baseline network is shown in Figure 2.7.

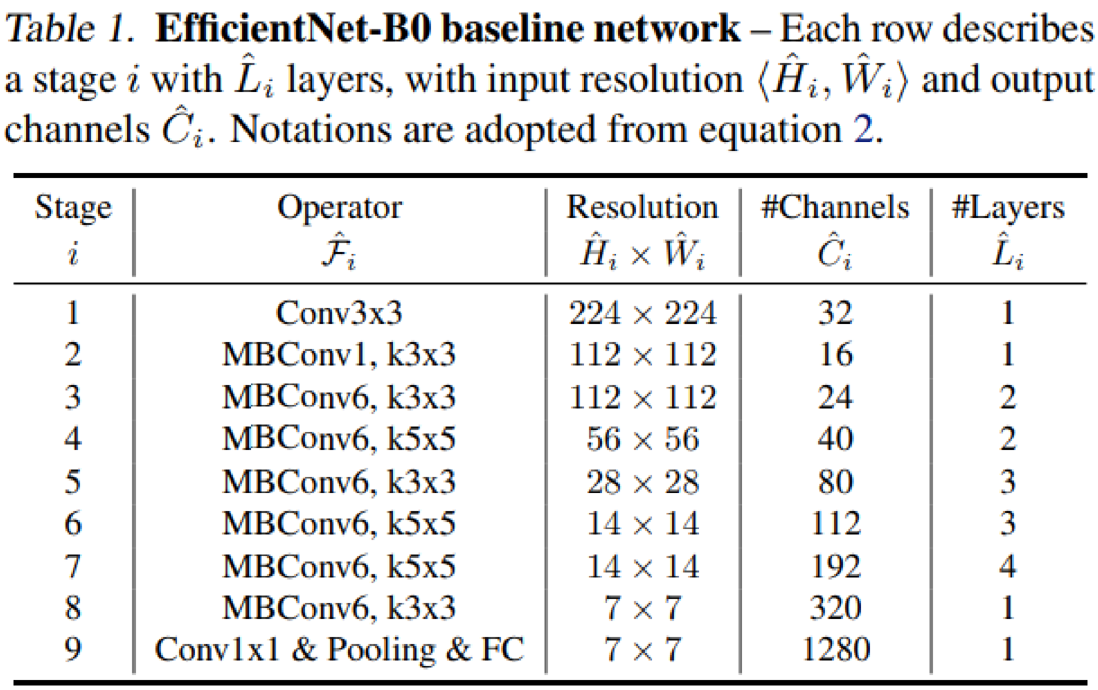


Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 2.7 Architecture of EfficientNet-B0.

By scaling up the dimensions uniformly, EfficientNet-B1 to B7 achieved higher accuracy scores compared to other models while at the same time has a lower number of parameters. For example, in EfficientNet-B0 the model can achieve a top-5 accuracy of 93.3% with only 5.3 million parameters, whereas DenseNet-169 has a top-5 accuracy of 93.2% but requires 14 million parameters. The result is even more profound when EfficientNet is scaled up from the baseline, EfficientNet-B1 achieved a top-5 accuracy of 94.4% with 7.8 million parameters while DenseNet-264 only achieved 93.9% but with 4.3 times the number of parameters. It is evident that by scaling up the dimensions uniformly, EfficientNet can perform significantly better and requires fewer parameters compared to other models. The authors have also explored the performance of EfficientNet on transfer learning. It is found that EfficientNet achieved better accuracy in transfer learning compared to other state-of-the-art models with a magnitude of 9.6 times fewer parameters.

CHAPTER 3 SYSTEM DESIGN AND METHODOLOGY

3.1 System Design

Figure 3.1 shows the flowchart of the proposed research.

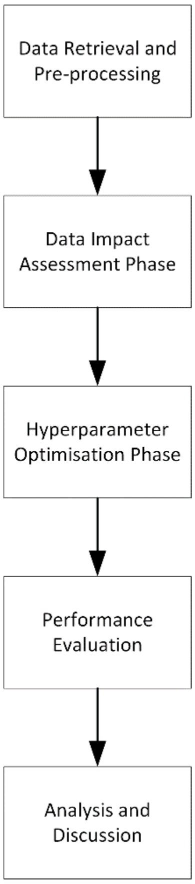


Figure 3.1 Flowchart of the proposed research.

After data retrieval and data pre-processing, this research is separated into two phases: data impact assessment phase and hyperparameter optimisation phase. It is then followed by model performance evaluation and result analysis.

3.1.1 Data Impact Assessment Phase

In this phase of our study, we conducted a comprehensive analysis to investigate how different types of wafer map data can impact the performance of CNN models. This analysis involved comparing the performance of the model using two types of data: the raw, unaltered wafer map data and data that had undergone pre-processing through connected component labelling, which served to highlight clusters that forms the distinct defect pattern of each class and reduce noise in the wafer maps. Additionally, we conducted a comparative evaluation of the model's performance by training it on both augmented and non-augmented datasets. Furthermore, we carried out a comparative assessment of the application of different pooling stratetgy within the classification layer of our model.

3.1.2 Hyperparameter Optimisation Phase

Fine-grained optimization of hyperparameters is carried out in this phase of the research. This iterative process involved systematically exploring variations in the number of densely connected layers, the number of hidden units, and the introduction of dropout as regularisation method. This is followed by different batch size configurations, initial learning rates (lr), and optimisers. The classification layer configurations and hyperparameter settings are outlined in Figure 3.7 and Table 3.3 respectively. Through this investigation, the optimal hyperparameter settings that yield peak performance was pinpointed and the finalised model obtained.

3.1.3 Performance Evaluation

At the end of the training process in each phase, a performance evaluation is conducted by testing the model on a separate test set. The evaluation is based on several key metrics, each providing valuable insights into the performance of the model.

1. Accuracy
    - Accuracy measures the ratio of number of correct predictions to total number of predictions.
    - Accuracy is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

    where TP (True Positive) is the number of corrected predicted positive instances, TN (True Negative) is the number of corrected predicted negative instances, FP (False Positive) is the number of actual negatives predicted as positives, and FN (False Negative) is the number of actual positives but predicted as negatives.
2. Confusion Matrix
    - A confusion matrix shows the counts of TP, TN, FP, and FN for each class following prediction by a model. It provides an overview of the model's performance on each class by comparing actual class labels with predicted class labels. Each row in the matrix represents the actual classes while each column represents the predicted classes. In multiclass classification, this matrix can be

quite large and may not be intuitive to interpret, especially when the class sizes are imbalanced [22]. To address this, normalised confusion matrix is often used for multiclass classification. A normalized confusion matrix scales the counts to represent proportions or percentages, each diagonal value represents the accuracy for each class, while off-diagonal values indicate misclassification rates.

3. Precision
   - Precision measures the ability of the model to correctly classify positive instances without misclassifying negatives as positives.
   - Precision is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

4. Recall (Sensitivity)
   - Recall quantifies the model's ability to correctly identify all positive instances within a class.
   - Recall is calculated as follows:

$$Accuracy = \frac{TP}{TP + FN} \tag{3}$$

5. F1 score:
   - F1 score measures the balance between precision and recall of a model.
   - F1 Score is calculated as follows:

$$Accuracy = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{4}$$

3.2 Methodology

Figure 3.2 shows an overview flow of the methodology.

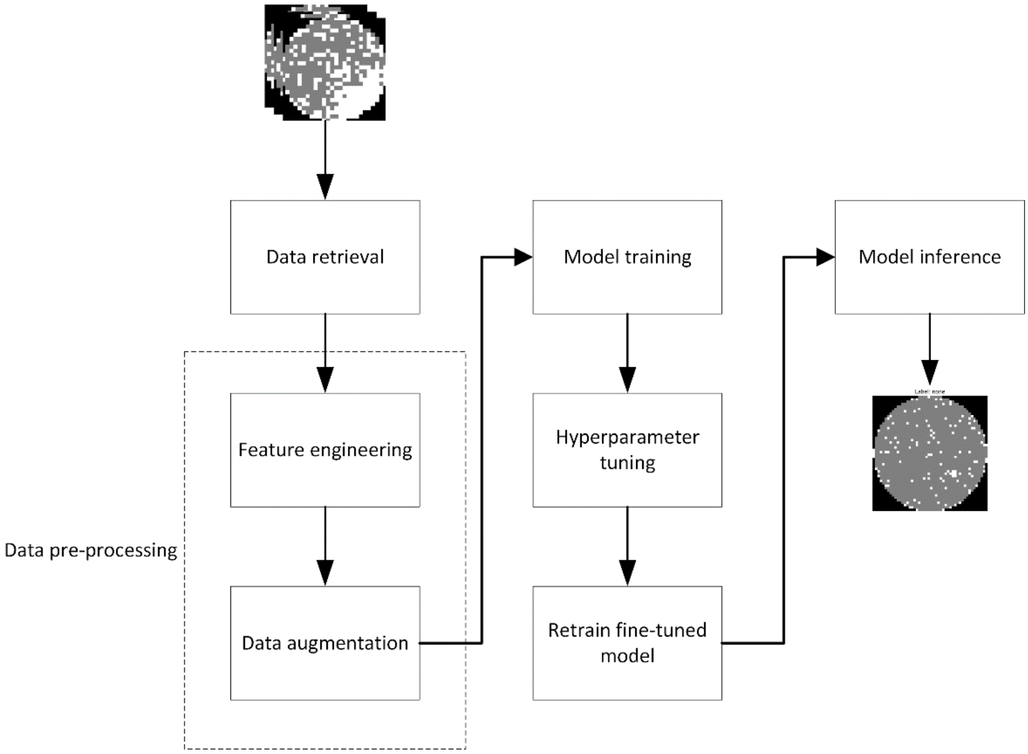


Figure 3.2 Pipeline of model training.

Before any classification task can be performed on the model, several publicly available wafer map datasets were considered for the evaluation of the model proposed. The real-world fabrication WM-811K wafer map dataset [9] would be the most suitable dataset to evaluate the model as it is extensively used by literature in the field and therefore can be used as a benchmark for performance comparison with previous works. The dataset consists of 811,457 rows and 6 columns.

The six columns in the dataset are as follows:

1. waferMap: 2-dimensional array that represents the wafer map. Each element in the array represents an individual die where a value of 0 represents a non-wafer area, a value of 1 represents a normal die, and a value of 2 represents a defective die.
2. dieSize: The size of each die in the wafer, measured in nanometres (nm).
3. lotName: The name or identifier of the manufacturing lot from which the wafer map was obtained.
4. waferIndex: The identifier of the wafer in the manufacturing lot.

5. trainTestLabel: A binary label indicating whether each wafer map is part of the training or test set.
6. failureType: The type of defect pattern in each wafer map, as labelled by domain experts. This is the target variable that we aim to classify.

There are 8 defect patterns and 1 normal class in the dataset:

1. Center
2. Donut
3. Edge-Loc
4. Edge-Ring
5. Loc
6. Near-full
7. Random
8. Scratch
9. None

In this study, we employed a preprocessing step aimed at enhancing the quality of the wafer map datasets before subjecting them to CNN classification. This preprocessing step involved highlighting the clusters that characterise the defect patterns and noise reduction using connected component labelling (CCL). CCL is a powerful technique used to identify and label distinct clusters within an image, making them more prominent and identifiable. Once the clusters are identified, the remaining noises are removed from the wafer images. By applying CCL, we aimed to improve the discrimination of distinct defect clusters within the wafer map, potentially aiding the CNN in achieving better classification accuracy.

Traditional methods for dealing with imbalanced datasets, such as random undersampling and oversampling, can lead to overfitting and reduced classification performance. Classifiers may fail to learn important features and patterns related to the majority class when undersampling, as some of the of the relevant information might be lost due to the removal whereas oversampling where replicated data is simply appended to the original dataset can lead to overfitting [23]. In the case of the wafer maps dataset, where there is a wide distribution of classes, undersampling the majority classes may remove defect patterns that is important. As shown in Figure 3.3, there may be multiple variants within a class, if undersampling is employed, some variants may not be included in the training process and may lead to poor classification performance.

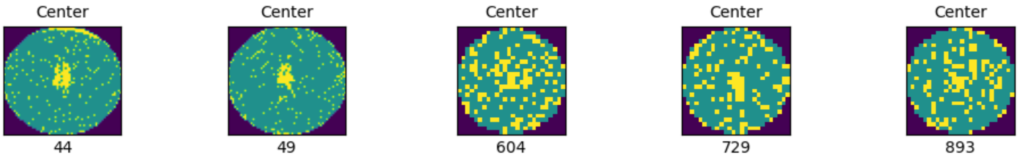


Figure 3.3 Samples showing different variants within Center class.

To address the class imbalance problem and to create robust feature representation of the wafer map, a method that uses a convolutional autoencoder to generate synthetic wafer map images for the minority classes were proposed. The autoencoder is trained with wafer maps from the original dataset and learns to reconstruct the images with minimal loss. We then generate synthetic wafer map images for the minority classes by sampling from the latent space of the encoder network and reconstruct the images with the decoder network. The synthetic wafer maps were then combined with the real wafer map images in the dataset to create a balanced dataset for model training. The synthetic wafer maps generated for the minority classes will then be merged with the down-sampled real wafer map images to construct a balanced dataset, consisting of around 7500 samples in each class.

A base model of EfficientNet-B0 variant is then setup with weights trained on ImageNet. The top classification layers of the pre-trained EfficientNet-B0 were frozen. This decision was made because we intended to employ this model primarily for feature extraction rather than classification. By doing so, we ensured that the previously learned representations and features extracted from ImageNet were preserved. A redesigned fully connected layer based on the number of classes of defect patterns as its output neuron will be implemented as the top classification layer of the EfficientNet-B0. With the redesigned classification layer included at the top of the EfficientNet-B0 model, training and fine-tuning will then be performed on the training set. The ratio of the train, validation and test set is set at 60:20:20. The rationale behind this specific split is to ensure an adequate representation of all classes within each subset. Given the presence of nine distinct classes, each with its own set of variations, the 60:20:20 ratio allows for a more comprehensive coverage of these variations in both the validation and test sets. When the classification result conducted on the training set is deemed satisfactory, model inference will be conducted using the trained model on the test set.

3.3 Model Architecture

3.3.1 Convolutional Autoencoder

To address the class imbalance problem in the dataset, we propose a convolutional autoencoder (CAE) as a data augmentation technique. Similar approaches have been used in previous studies to address class imbalance in the WM-811K dataset [24], [25]. The CAE is designed to learn a compressed representation of the wafer maps and can be used to generate synthetic data that can be added to the original dataset to balance the classes.

The CAE consists of an encoder and a decoder, with the encoder having one convolutional layer with 64 filters followed by a max pooling layer. The decoder consisted of two transpose convolutional layers with 64 and 3 filters, respectively, each followed by an up-sampling layer. The final layer of the decoder had a sigmoid activation function to generate the output. The CAE was then trained using the mean squared error loss function and the Adam optimizer for 15 epochs with a batch size of 1024. The architecture of the CAE is shown in Figure 3.4.

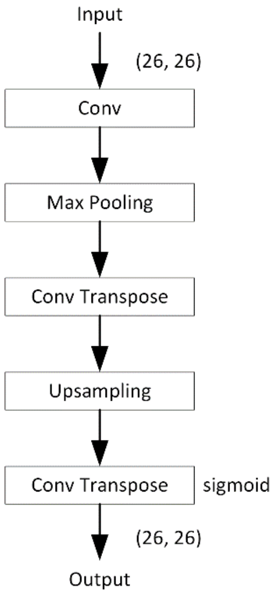


Figure 3.4 Convolutional autoencoder architecture

3.3.2 EfficientNet-B0 Model Architecture

Figure 3.5 shows the architecture of EfficientNet-B0 model.

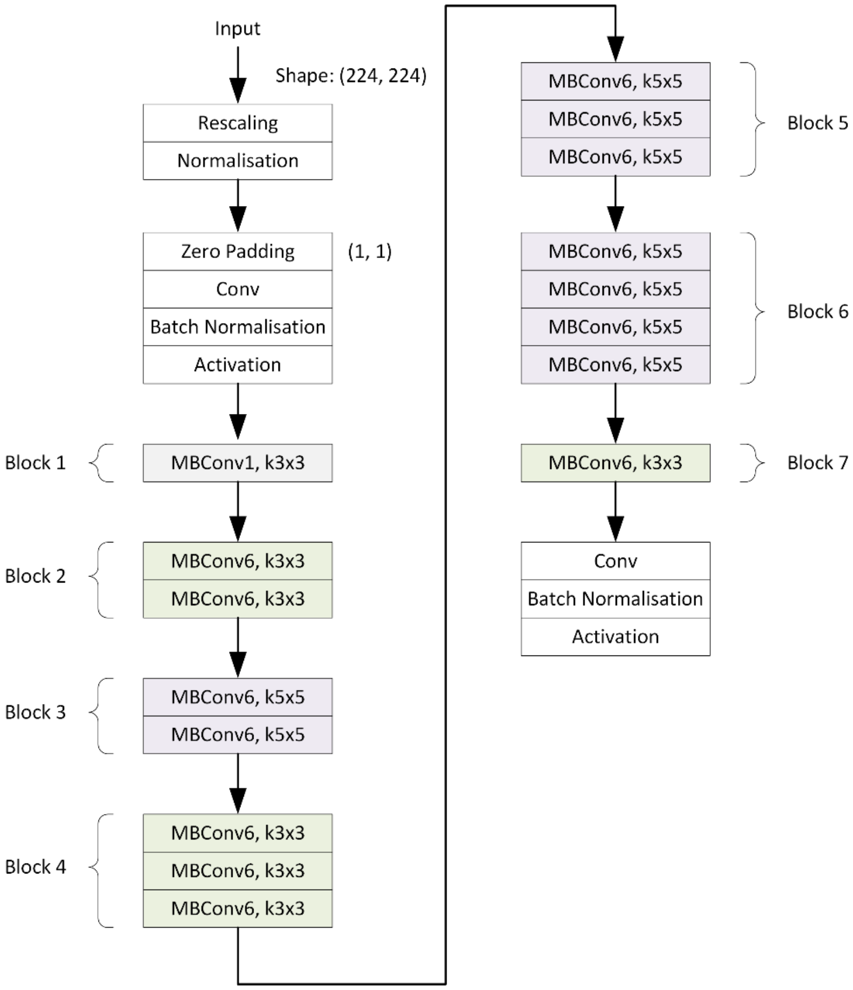


Figure 3.5 Architecture diagram of EfficientNet-B0.

In the EfficientNet-B0 model loaded using Keras, the architecture of the model consists of the input layer, an input pre-processing layer, a stem layer, 7 blocks of mobile inverted bottleneck MBConv layers, and a top layer.

EfficientNet-B0 model expects the image inputs to be in the [0-255] range. In the EfficentNet-B0 model loaded with Keras, the input pre-processing has already been included using a pass-through function. The pre-processing layer includes rescaling the pixel values from range [0, 255] to the range [0, 1] and a normalisation layer to centre and scale the mage to have zero mean and unit variance.

In the stem layers, zero padding pads the input tensor with zeros to ensure that the input has the correct spatial dimensions for the subsequent convolutional layer. A 2D convolution is performed on the input tensor, with a kernel size of 3×3, a stride of 2, and 32 output filters. A batch normalisation layer then normalises the activations of the previous convolutional layer across the batch dimension, and then scales and shifts the normalised values using learned parameters. An activation layer then applies non-linear function elementwise to the output of the batch normalisation layer.

Each MBConv block consists of the following layers:

1. DepthwiseConv2D: This layer performs a depthwise convolution on the input tensor, which means that it applies a separate convolutional filter to each input channel, without mixing the channels. The kernel size and stride of the convolutional filter are determined by the hyperparameters of the block.
2. BatchNormalization: This layer normalizes the activations of the depthwise convolutional layer across the batch dimension, and then scales and shifts the normalized values using learned parameters. This helps to improve stability and prevent overfitting during training.
3. Activation: This layer applies the activation function to the output of the previous batch normalization layer.
4. Add: This layer adds the output of the previous activation layer to the input tensor of the block, which creates a "residual connection" that allows the network to learn residual functions and reduce the vanishing gradient problem.
5. Squeeze: This layer aggregates the channel-wise feature responses of a convolutional layer into a single scalar value, which captures the relative importance of each channel in the feature map.
6. Excitation: This layer takes these scalar values and produces a set of channel-wise scaling factors, which are used to amplify the important channels and suppress the unimportant ones.

In the top layer, there is a 2D convolution layer, a batch normalisation layer, and an activation layer. Since we are modifying the top classification layer to classify nine classes, the default top classification layer preloaded with Keras are not included in the base model.

3.3.3 Data Impact Assessment Phase Classification Layer

The classification layer, depicted in Figure 3.6 were considered for the data impact assessment phase.

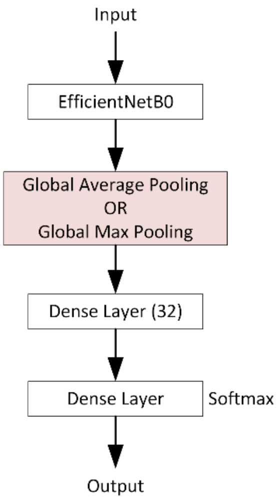


Figure 3.6 Classification layer for data impact assessment phase.

To establish a baseline and effectively analyse the impact of different data types on a model’s performance, only one classification layer configuration was considered for this phase. The classification layer consists of a global average pooling (GAP) layer and a Dense layer with 32 neurons with ReLU activation function to introduce non-linearity. The final Dense layer, containing 9 units, employs a softmax activation function to produce class probabilities for the multi-class classification task.

To illustrate the distinction between employing GAP and global max pooling (GMP) in our use case, we also assessed the classification performance using GMP in place of GAP in the classification layer configuration. GAP computes the mean value of all elements within each feature map, effectively reducing the spatial dimensions (height and width) of each map to a 1×1 size. Consequently, a single value per feature map is produced, which is then aggregated to create a fixed-size output vector [26]. In contrast, GMP identifies the maximum value present in each feature map, while also reducing the spatial dimensions of each map to 1×1 as well. The resulting maximum values are subsequently combined to form a fixed-size output vector. The primary distinction between GAP and GMP lies in their sensitivity to spatial information. GAP is beneficial for capturing the overall presence of features across the entire feature map, while GMP provide a more localized representation of spatial information.

3.3.4 Hyperparameter Optimisation Phase Classification Layers

The different classification layers, depicted in Figure 3.7 were considered for the hyperparameter optimisation phase.

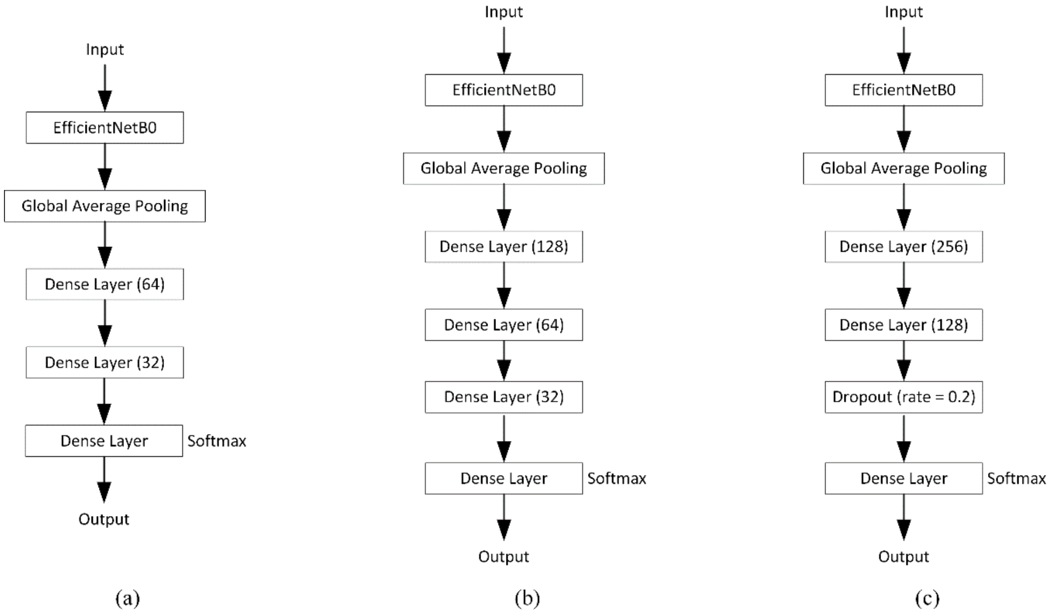


Figure 3.7 Classification layers for hyperparameter optimisation phase.

In the hyperparameter optimisation phase, we consider three classification layer configurations. The configurations are as follows:

1. GAP with two Dense layers
    - This configuration comprised a GAP layer, followed by two fully connected Dense layers with 128 and 64 units, respectively. ReLU activation functions were used in both Dense layers.
    - We chose this configuration as a baseline to establish a starting point for our analysis. The inclusion of two Dense layers with decreasing units allowed for the introduction of non-linearity, while GAP captured general spatial information across feature maps. This configuration provided a balanced approach between complexity and simplicity.
2. GAP with three Dense layers
    - In this configuration, we retained the GAP layer followed by three fully connected Dense layers with 128, 64, and 32 units, respectively. ReLU activation functions were applied to each Dense layer.

  - We expanded upon the first configuration by adding an additional Dense layer, increasing the model's capacity to capture complex patterns in the data. This deeper architecture was intended to explore whether increased depth would lead to improved classification performance.

3. GAP with additional dropout
    - This configuration incorporated GAP followed by two Dense layers with 256 and 128 units, respectively. Notably, a dropout layer with a dropout rate of 0.2 was inserted between the second and third Dense layers to introduce regularization.
    - Due to the increasing depth of the Dense layers, we aimed to assess the impact of dropout regularisation on classification performance. Our intention was to enhance the model's ability to generalize and exhibit robustness as a result of the increased layer depth by incorporating dropout in the classification layers to prevent the model from overfitting.

3.4 Hyperparameter Settings

3.4.1 Data Impact Assessment Phase Hyperparameter Settings

In the data impact assessment phase, to establish a baseline, a fixed set of hyperparameters was employed as shown in Table 3.1.

Hyperparameter	**Value/Description**
Number of epochs	20
Learning rate	0.001
Learning rate scheduler	Decay by a factor of 0.5 every 10 epochs
Batch size	32
Optimiser	Adam

Table 3.1 Hyperparameter setting for data impact assessment phase.

3.4.2 Hyperparameter Optimisation Phase Hyperparameter Settings

In the hyperparameter optimisation phase, we divided our hyperparameter settings into two distinct parts as follows:

1. Fixed Hyperparameter Settings
   In this initial part, we focused on optimising the classification layer configurations specified in Section 3.3.4 while keeping other critical hyperparameters constant. The fixed hyperparameters used are shown in Table 3.2.

Hyperparameter	**Value/Description**
Number of epochs	40
Learning rate	0.001
Learning rate scheduler	Discrete staircase decay by a factor of 0.5 every 10 epochs
Early stopping	Patience: 10 Monitor: Validation loss
Batch size	32
Optimiser	Adam

Table 3.2 Hyperparameter optimisation phase fixed hyperparameter settings.

2. Hyperparameter Tuning
   Once we obtained an optimal classification layer architecture, we transitioned to hyperparameter tuning. Here, we systematically explored a range of hyperparameters to further optimize the model's performance. The hyperparameters under consideration is shown in Table 3.3.

Hyperparameter	**Value/Description**
Number of epochs	40
Learning rate scheduler	Discrete staircase decay by a factor of 0.5 every 10 epochs
Early stopping	Patience: 10 Monitor: Validation loss
Batch size	**16**
	32
	64
Initial learning rate	**0.01**
	0.001
	0.0001
Optimiser	**Adam**
	RMSprop
	SGD

Table 3.3 Hyperparameter optimisation phase hyperparameter tuning.

3.5 Tools and Technologies

3.5.1 Software

Python is programming language used to train the proposed deep learning model in this project. Jupyter Lab and Google Colab will be the preferred IDEs for this project due to its ease of configuration and easy sharing. Keras, an open-source API for deep learning models development in Python will be the preferred software library for this project. Additional libraries such as NumPy and pandas will be used as well. Matplotlib and seaborn will also be used to visualise graphs in the project.

3.5.2 Hardware

The hardware specifications used in this research is shown in Table 3.4 and Table 3.5.

Description	**Specifications**
OS	Windows 10 Home 64-bit
CPU	AMD Ryzen 7 5800H
System RAM	16.0 GB
GPU	Nvidia GeForce RTX 3060 1.32 GHz Base Clock 6GB GDDR6 Memory

Table 3.4 Local hardware specifications.

Description	**Specifications**
CPU	Intel Xeon 2vCPU @ 2.2GHz
System RAM	83.5 GB
GPU	Nvidia Tesla A100 / Nvidia Tesla V100 40GB VRAM

Table 3.5 Google Colab hardware specifications.

CHAPTER 4 EXPERIMENT RESULTS AND DISCUSSION

4.1 Data Retrieval and Exploration

Prior to initiating the training and development of the classification model, a preliminary exploration of the dataset is essential to acquire a comprehensive understanding of the underlying data.

The dataset used in this study was sourced from the paper “Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets" by C. J. Hwang et. al. [9]. Their work involved testing wafer map defect pattern on model that is rotation- and scale invariant led to the creation of the world's largest publicly accessible dataset of wafer maps, which comprises 811,457 real-world wafer maps.

The dataset contained 811,457 instances with six columns, namely waferMap, dieSize, lotName, waferIndex, trainTestLabel, and failureType. The waferMap column contains 2D representations of the wafer stored in a 2D array where a value of 0 denotes non-wafer areas, 1 indicates non-defective dies and 2 represents defective dies. Further exploration of the dataset revealed that there were eight different types of failure patterns: Center, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, and Near-full. Figure 4.1 provides a visual representation of the eight distinct failure patterns identified in the dataset. Each pattern is represented by a color-coded image, with the corresponding failure type labelled at the top of each image.

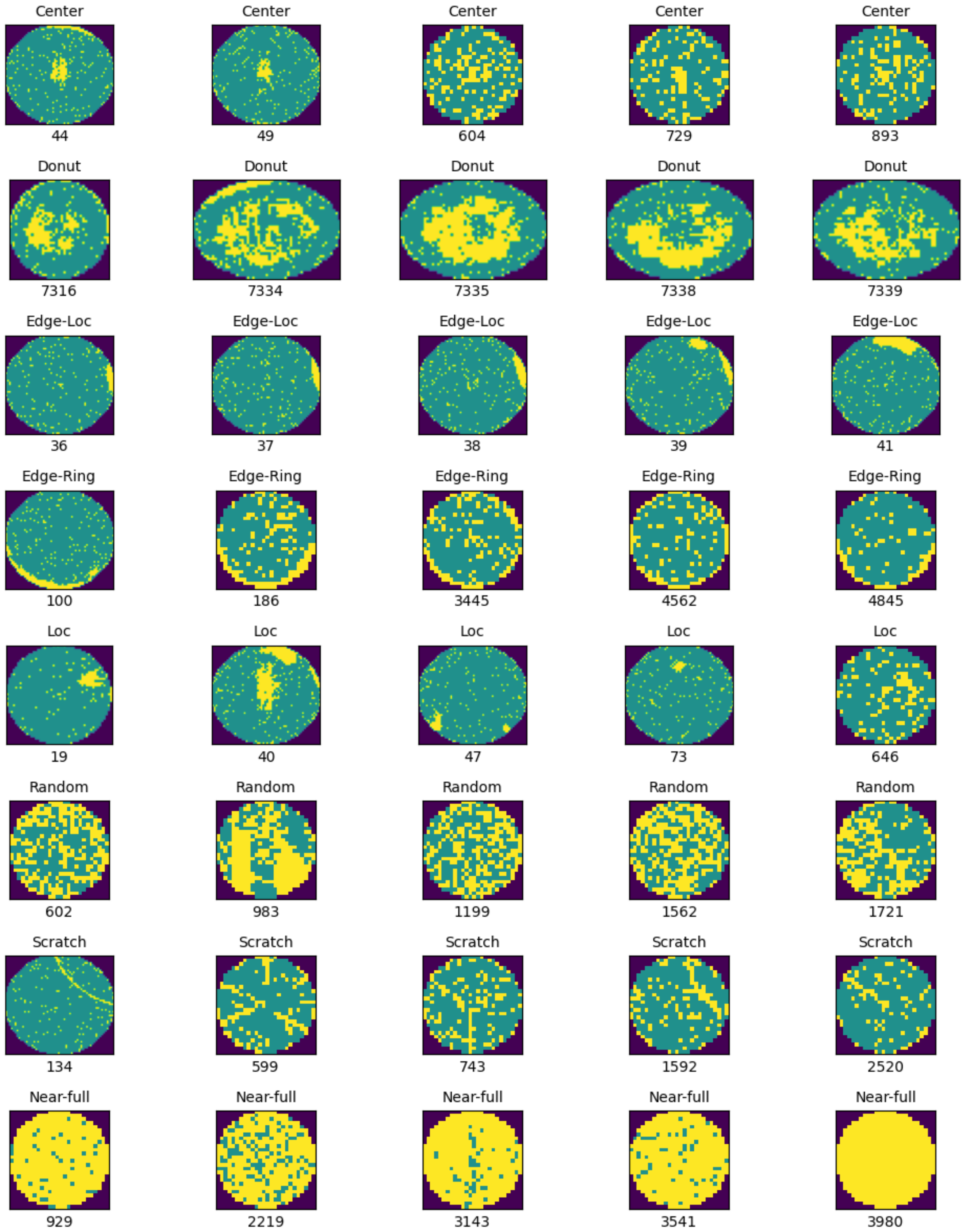


Figure 4.1: Visualization of Wafer Failure Patterns: Examples of eight distinct failure patterns observed in the dataset, including Centre, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, and Near-full.

Furthermore, an analysis of the wafer map dimensions was conducted to identify the range of dimensions within the dataset. The minimum wafer map dimension was found to be (22, 35), while the maximum was (212, 84), indicating a significant range of wafer map sizes in the dataset. CNNs often require input data to have a fixed size, especially in the case of image data so that they can be processed in a consistent manner. This is because the number of trainable parameters in the network is dependent on the input shape where varying input sizes would require different number of parameters.

Through further exploration, it was discovered that a significant portion of the dataset (78.7%) comprised of unlabelled wafers, while only a small proportion of wafers (3.1%) had been labelled with the eight failure patterns. Additionally, 18.2% of the wafers contained no failure patterns, as depicted in Figure 4.2.

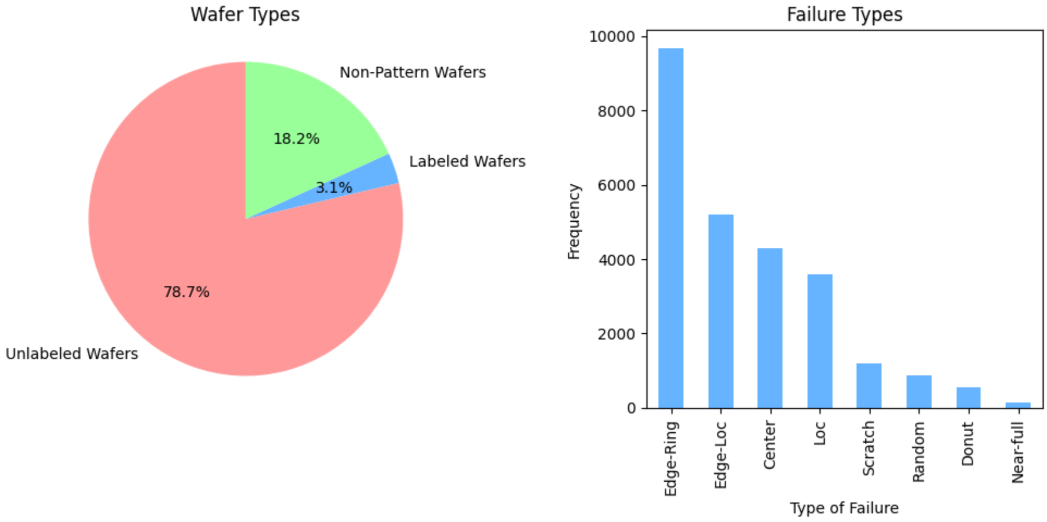


Figure 4.2 Pie chart and bar plot of wafer dataset distribution: Proportions of labelled and non-labelled wafers with frequency of failure types.

Each instance of labelled wafer maps in the dataset are also assigned as train or test by domain experts. The train set consists of wafer maps that include each pattern type while the test set were randomly selected by domain experts [9]. The distribution of failure types for both train and test data are shown in Figure 4.3. Based on the distribution of failure types in the train data, it can be inferred that there is a significant imbalance in the frequency of occurrence of different types of failures. Most of the wafers in the train data belong to the Edge-Ring failure type, which constitutes 58.5% of the labelled data. On the other hand, the Near-full failure type has the least representation, comprising only 0.7% of the labelled data. This uneven distribution of failure types could potentially affect the performance of the classification model.

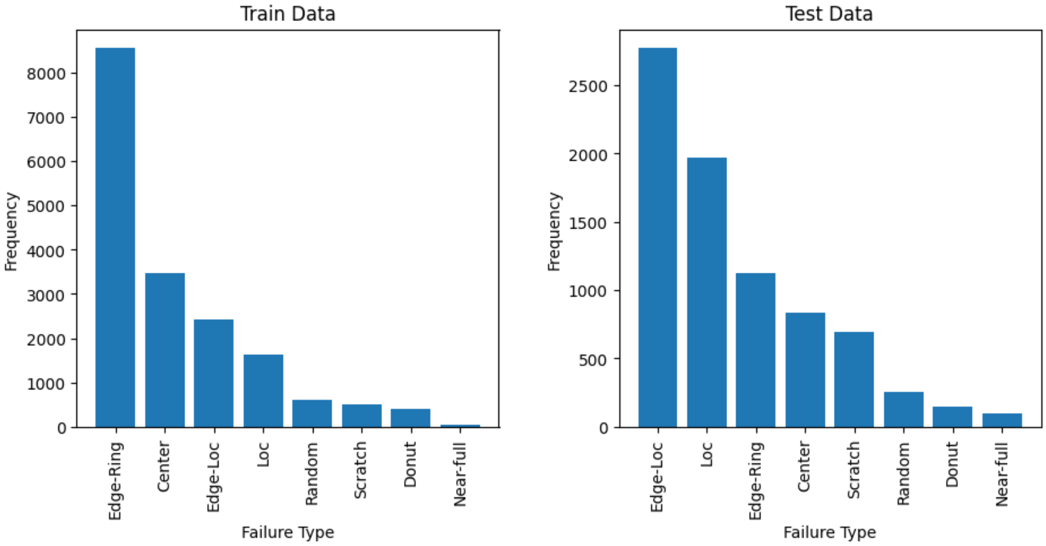


Figure 4.3 Distribution of different classes in train and test data.

4.2 Data Pre-processing

4.2.1 Cluster Highlighting and Noise Reduction

In the initial pre-processing phase, we implemented a procedure to enhance the quality of the wafer map datasets. This pre-processing step was specifically designed to address two key objectives: highlighting the underlying defect clusters and mitigating noise within the data.

We initiated the process by casting all instances of value ‘1’ to ‘0’, leaving only value ‘2’, which denotes defective dies, on the wafer map data. CCL with 8-connected neighbourhood were then performed on the wafer map data, effectively highlighting any clusters. To ensure that only meaningful clusters were emphasised, a cluster size threshold was established. Clusters containing less than 15 data points were disregarded, as theses were deemed too small to constitute significant clusters that represents the defect patterns. Clusters containing more than 15 data points were assigned the value ‘3’. Following cluster highlighting, any remaining ‘2’ values, which were considered noise, were reassigned to value ‘1’.

The original and denoised samples for each defect patterns are shown in Figure 4.4.

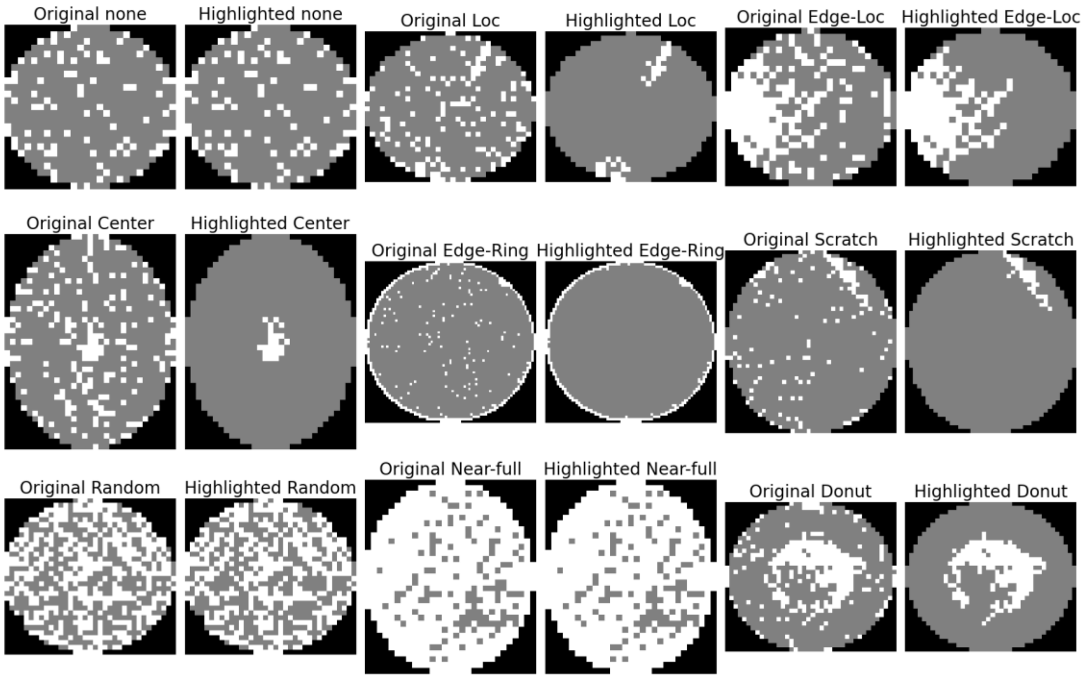
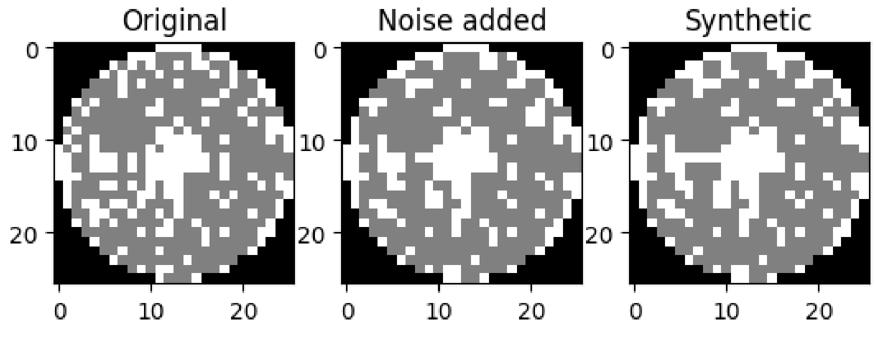


Figure 4.4 Original and denoised samples.

4.2.2 Data Augmentation

To generate the synthetic wafer map data, the CAE was used to encode and then decode the wafer maps. For each minority class in the train set, we augmented the number of synthetic samples for each minority class until a sufficient sample count was achieved. To create the synthetic samples, wafer maps from the minority classes were selected and encoded using the trained CAE. Gaussian noise was then added to the encoded features to introduce variability, and the noisy encoded features were decoded to generate synthetic wafer maps. The generated wafer maps and their corresponding label were then added to the original dataset. Figure 4.5 shows the augmentation process on a wafer map using the CAE.



(a) (b) (c)

Figure 4.5 (a) Original; (b) Encoded wafer map with Gaussian noise added; (c) Decoded wafer map.

The number of samples before and after augmentation is shown in Table 4.1.

	Number of samples			
Class	Before Augmentation	Training Set (80%)	Augmented Training Set	Test Set (20%)
Center	4294	3435	7743	859
Donut	555	444	7548	111
Edge-Loc	5189	4151	8302	1038
Edge-Ring	9680	7744	7744	1936
Loc	3593	2874	7743	719
Near-full	149	119	7735	30
None	9000	7200	7200	1800
Random	866	693	7623	173
Scratch	1193	955	7640	238
Total	**34519**	**27615**	**69278**	**6904**

Table 4.1 Number of samples before and after augmentation

It is important to note that in our research, we opted to reduce the 'None' class from 147,431 samples to 9,000 samples. This decision was made considering that the 'None' class lacks the internal variations present in other classes. Therefore, downsizing this class's sample count to align with the subsequent majority class, 'Edge-Ring,' was deemed appropriate.

4.2.3 Image Resizing

When utilising pre-trained models for wafer map classification, all wafer map images must be resized to a fixed size. Pre-trained models are trained on large-scale datasets, such as ImageNet, to extract features from images and the features extracted can then be used as input for a downstream task. However, pre-trained models require input images to have a fixed size. For example, in EfficientNet-B0, the pretrained baseline variant of EfficientNet [16], requires the input image to be 224×224. Therefore, the wafer maps in the dataset that contains the original wafer maps and synthetic wafer maps generated by the CAE was resized to a fixed size of 224×224. By resizing wafer maps in the dataset to a fixed size ensures that they can be effectively processed by the pre-trained model and that the extracted features can be accurately used for classification.

There are several image scaling methods such as nearest-neighbour, bilinear interpolation, bicubic interpolation, and B-spline interpolation. While several findings have shown that nearest-neighbour yields the worst result when it comes to scaling images when evaluated with other methods [27], [28], nearest-neighbour interpolation is the best for this use case. This is because the pixels in the wafer maps are represented by only three different values, 0s, 1s, and 2s. This means that the values of the pixels represent discrete features that should not be interpolated or averaged more than necessary to preserve the features. Nearest neighbour interpolation selects the nearest pixel value to the new pixel location, making it the most appropriate interpolation method for the wafer map image. Other interpolation methods, such as bilinear or bicubic interpolation can introduce artificial pixel values and blur the boundaries between the individual pixels in the image which can lead to misclassification in deep learning models. Based on the observations in Figure 4.6, we can see that the application of bilinear and bicubic interpolation during the image scaling process causes the individual pixel features of the original image, which correspond to dies in the wafer map, to be blurred and distorted. Whereas image scaling using nearest-neighbour interpolation preserves these important features.

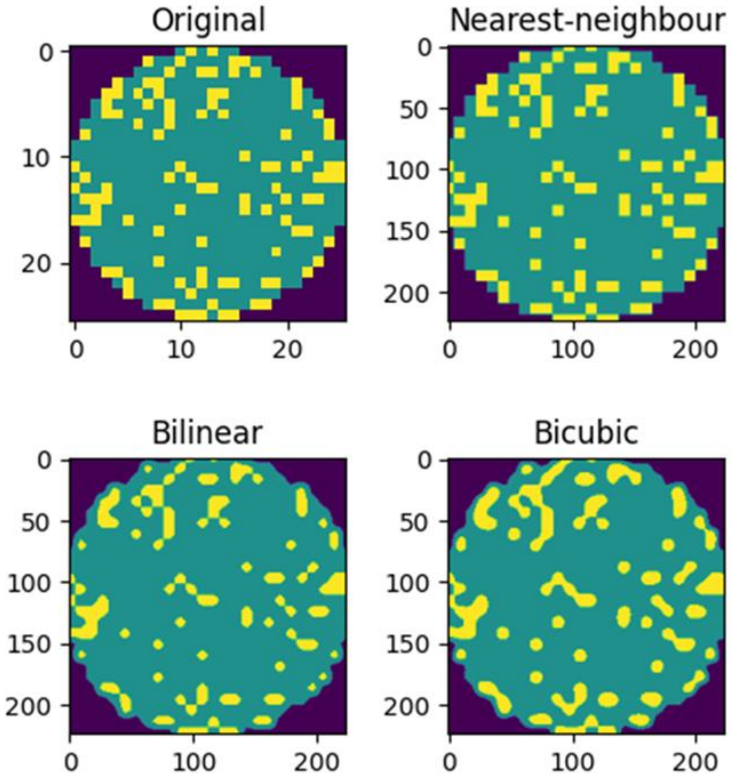


Figure 4.6 Comparison of different resizing methods on a wafer map sample: Nearest-neighbour, bilinear, and bicubic interpolation.

4.3 Data Impact Assessment Phase Model Training and Evaluation

4.3.1 Training Results

In the data impact assessment phase, four variations of the wafer map image dataset were considered: raw, unaltered data, noise reduction pre-processed data, non-augmented data, and augmented data. Additionally, we also train the models using different pooling strategies, namely global average pooling and global max pooling within this phase. In total, there are 4 model schemes experimented in this phase, namely raw (RU), noise reduction/non-augmented (NR-NA), noise reduction/augmented using global average pooling (NR-A-GAP), and noise-reduction/augmented using global max pooling (NR-A-GMP). The specific hyperparameter settings employed during this phase are detailed in Section 3.4.1.

In Figure 4.7, Figure 4.8, and Figure 4.9, we present the accuracy and loss curves for training on each dataset variation.

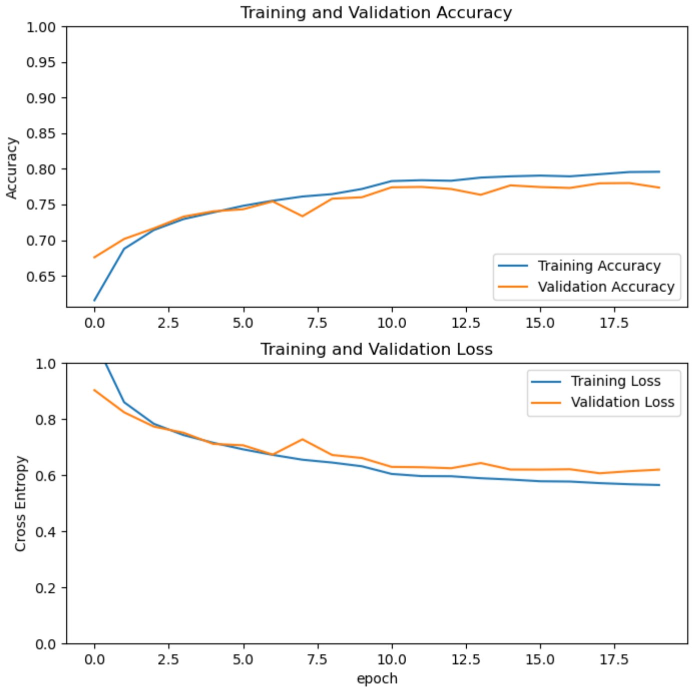


Figure 4.7 Accuracy and loss curves for training on RU model.

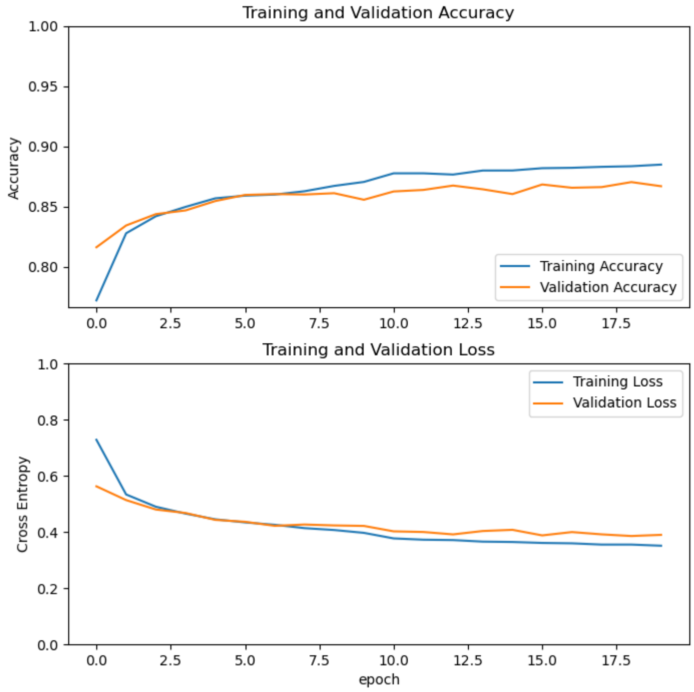


Figure 4.8 Accuracy and loss curves for training on NR-NA model.

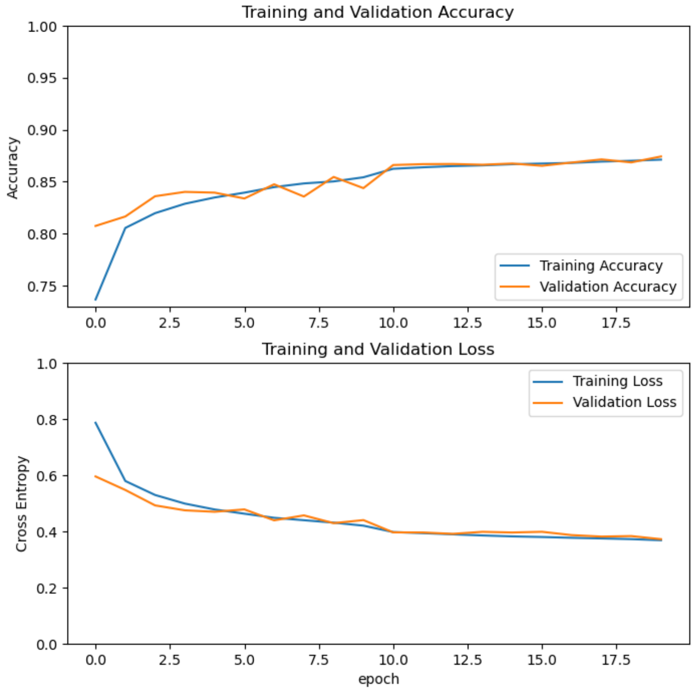


Figure 4.9 Accuracy and loss curves for training on NR-A-GAP model.

Comparing the accuracy and loss between the RU model and the NR-NA model, we can see that the model trained on the noise reduction pre-processed dataset (NR-NA) exhibited higher validation accuracy and lower loss values compared to the raw, unaltered dataset (RU) at the end of training. This indicates that the model trained on the noise reduction pre-processed dataset benefitted from the improved data quality and reduced susceptibility to noise that led to better generalisation.

When comparing the accuracy and loss between the NR-NA model and NR-A-GAP model, it is evident that the non-augmented dataset (NR-NA) suffers from slight overfitting, where the validation accuracy and loss values begins to plateau starting from the 8th epoch onwards, whereas the training accuracy and loss values continue to improve. In the model trained on augmented data (NR-A-GAP), the validation accuracy and loss values continue to improve along with the training values. This shows that by increasing the size of the dataset and balancing the class distribution, the model has a better chance of capturing the underlying patterns in the data.

4.3.2 Model Evaluation

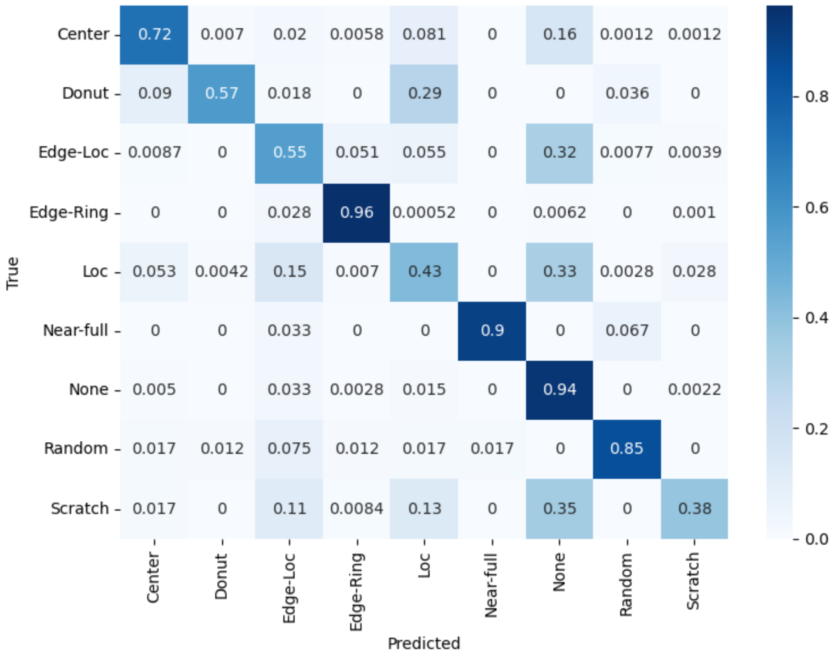
After the models are trained, we evaluate them on the test set using the metrics specified in Section 3.1.3.Table 4.2 shows the accuracy of each model.

Model	RU	NR-NA	**NR-A-GAP**	NR-A-GMP
Accuracy	0.7816	0.8659	**0.8766**	0.8599

Table 4.2 Accuracy values for models trained on each dataset variation and pooling variation.

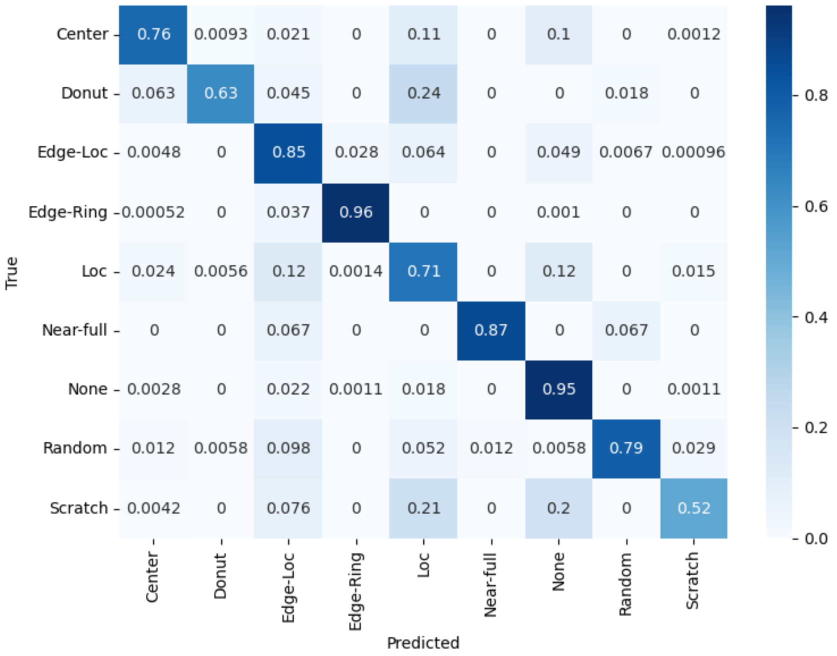
We can see that the model trained with noise reduction/augmented data and a global average pooling layer demonstrated higher accuracy compared to others. However, accuracy alone may not provide a complete assessment for multiclass classification, where a high accuracy score can sometimes mask the model's ability to distinguish between different classes effectively.

Figure 4.10, Figure 4.11, Figure 4.12, and Figure 4.13 show the normalised confusion matrix for each model.



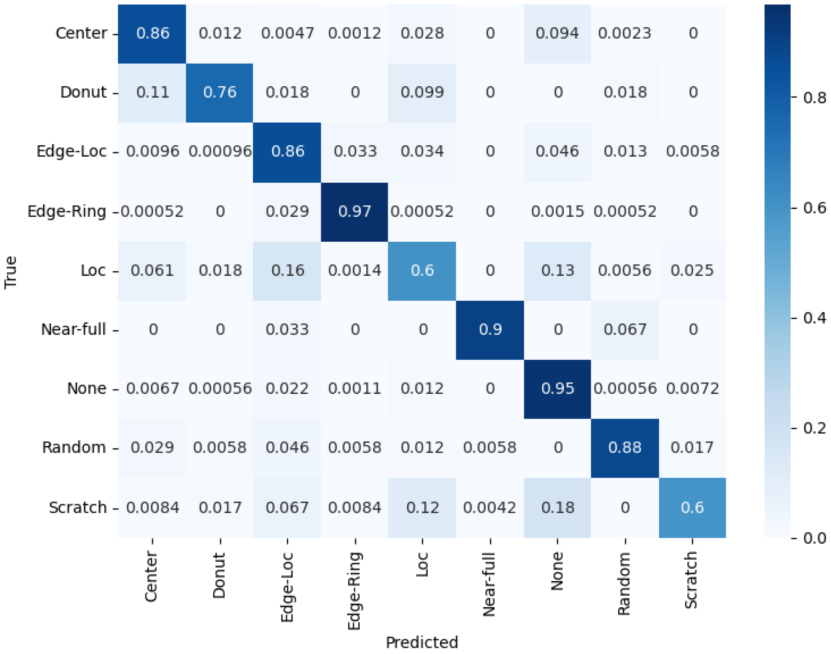
	Center	Donut	Edge-Loc	Edge-Ring	Loc	Near-full	None	Random	Scratch
Center	0.72	0.007	0.02	0.0058	0.081	0	0.16	0.0012	0.0012
Donut	0.09	0.57	0.018	0	0.29	0	0	0.036	0
Edge-Loc	0.0087	0	0.55	0.051	0.055	0	0.32	0.0077	0.0039
Edge-Ring	0	0	0.028	0.96	0.00052	0	0.0062	0	0.001
Loc	0.053	0.0042	0.15	0.007	0.43	0	0.33	0.0028	0.028
Near-full	0	0	0.033	0	0	0.9	0	0.067	0
None	0.005	0	0.033	0.0028	0.015	0	0.94	0	0.0022
Random	0.017	0.012	0.075	0.012	0.017	0.017	0	0.85	0
Scratch	0.017	0	0.11	0.0084	0.13	0	0.35	0	0.38

Figure 4.10 Confusion matrix for RU model.



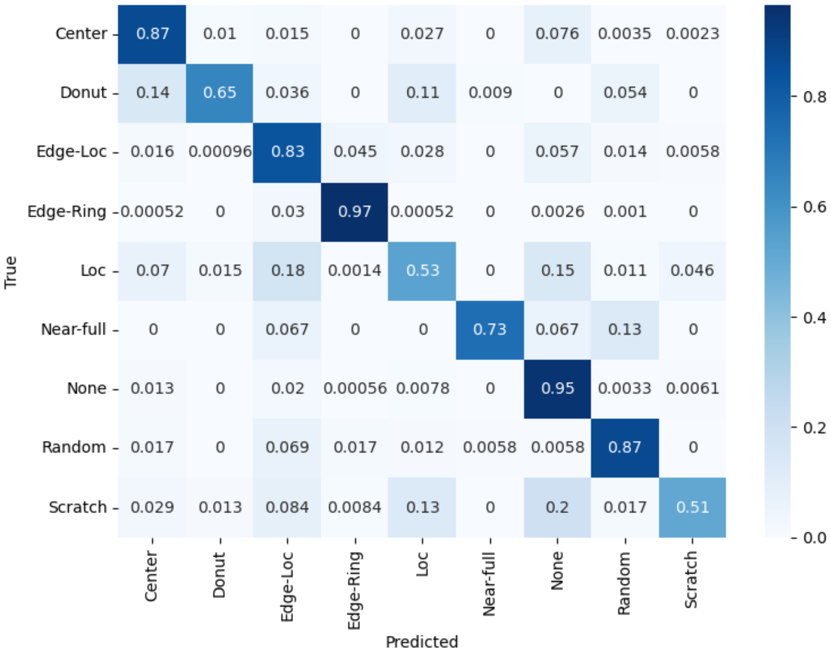
	Center	Donut	Edge-Loc	Edge-Ring	Loc	Near-full	None	Random	Scratch
Center	0.76	0.0093	0.021	0	0.11	0	0.1	0	0.0012
Donut	0.063	0.63	0.045	0	0.24	0	0	0.018	0
Edge-Loc	0.0048	0	0.85	0.028	0.064	0	0.049	0.0067	0.00096
Edge-Ring	0.00052	0	0.037	0.96	0	0	0.001	0	0
Loc	0.024	0.0056	0.12	0.0014	0.71	0	0.12	0	0.015
Near-full	0	0	0.067	0	0	0.87	0	0.067	0
None	0.0028	0	0.022	0.0011	0.018	0	0.95	0	0.0011
Random	0.012	0.0058	0.098	0	0.052	0.012	0.0058	0.79	0.029
Scratch	0.0042	0	0.076	0	0.21	0	0.2	0	0.52

Figure 4.11 Confusion matrix for NR-NA model.



	Center	Donut	Edge-Loc	Edge-Ring	Loc	Near-full	None	Random	Scratch
Center	0.86	0.012	0.0047	0.0012	0.028	0	0.094	0.0023	0
Donut	0.11	0.76	0.018	0	0.099	0	0	0.018	0
Edge-Loc	0.0096	0.00096	0.86	0.033	0.034	0	0.046	0.013	0.0058
Edge-Ring	0.00052	0	0.029	0.97	0.00052	0	0.0015	0.00052	0
Loc	0.061	0.018	0.16	0.0014	0.6	0	0.13	0.0056	0.025
Near-full	0	0	0.033	0	0	0.9	0	0.067	0
None	0.0067	0.00056	0.022	0.0011	0.012	0	0.95	0.00056	0.0072
Random	0.029	0.0058	0.046	0.0058	0.012	0.0058	0	0.88	0.017
Scratch	0.0084	0.017	0.067	0.0084	0.12	0.0042	0.18	0	0.6

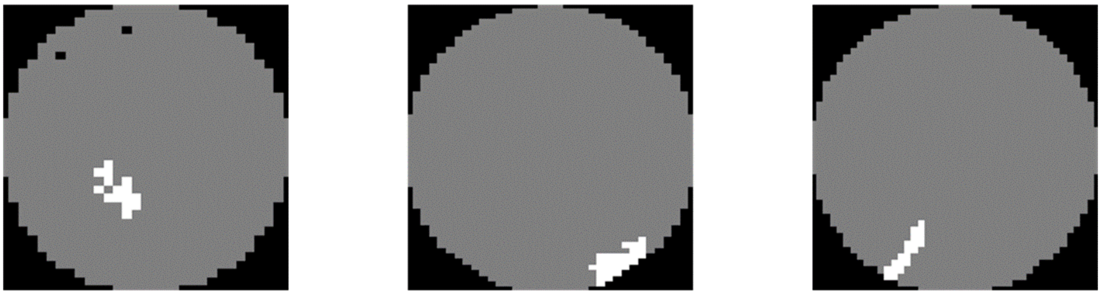
Figure 4.12 Confusion matrix for NR-A-GAP model.



	Center	Donut	Edge-Loc	Edge-Ring	Loc	Near-full	None	Random	Scratch
Center	0.87	0.01	0.015	0	0.027	0	0.076	0.0035	0.0023
Donut	0.14	0.65	0.036	0	0.11	0.009	0	0.054	0
Edge-Loc	0.016	0.00096	0.83	0.045	0.028	0	0.057	0.014	0.0058
Edge-Ring	0.00052	0	0.03	0.97	0.00052	0	0.0026	0.001	0
Loc	0.07	0.015	0.18	0.0014	0.53	0	0.15	0.011	0.046
Near-full	0	0	0.067	0	0	0.73	0.067	0.13	0
None	0.013	0	0.02	0.00056	0.0078	0	0.95	0.0033	0.0061
Random	0.017	0	0.069	0.017	0.012	0.0058	0.0058	0.87	0
Scratch	0.029	0.013	0.084	0.0084	0.13	0	0.2	0.017	0.51

Figure 4.13 Confusion matrix for NR-A-GMP model.

From the confusion matrices, we can see that NR-NA model performs better than the RU model, this shows that the noise reduction pre-processing step likely removed irrelevant information, which are the random defective dies in the wafer, resulting in easier identification of meaningful patterns. This allows the model to better distinguish between classes and make more accurate predictions, particularly on the ‘Donut’, ‘Edge-Loc’, ‘Loc’, and ‘Scratch’ classes. Similarly, by introducing augmented data and balanced distribution of classes, further improvements in classification performance can be seen, especially on ‘Center’, ‘Donut’, ‘Random’, and ‘Scratch’ class. However, deterioration of performance when using augmented data can be seen in the ‘Loc’ and ‘Scratch’ classes, where the NR-A models sometimes wrongly classified as ‘Edge-Loc’ and ‘Scratch’. This is due to the close similarity between these classes, as shown in Figure 4.14.



Loc Edge-Loc Scratch

Figure 4.14 Similarity in ‘Loc’, ‘Edge-Loc’, and ‘Scratch’ class.

The key distinguishing feature for these classes lies in their spatial placement within the image, especially ‘Loc’ and ‘Edge-Loc’. ‘Edge-Loc’ is characterised by a point of interest positioned near the wafer’s edge, whereas ‘Loc’ does not. These spatial variations become critical discriminative factors for these two classes. Unfortunately, CNNs, which are renowned for their translation invariance, may struggle to effectively capture these nuances. While the translation invariance property of CNNs enable them to recognise patterns and features irrespective of their position within an image, which can be advantageous for many computer vision tasks where the location of the object is not important, it can be limiting in scenarios where precise spatial information is crucial for classification. In our case, this limitation becomes evident as the network fail to discern between these classes due to their differing spatial characteristics.

Table 4.3, Table 4.4, and Table 4.5 shows the precision, recall, and F1 score value for each model.

	Precision			
Class	RU	NR-NA	**NR-A-GAP**	NR-A-GMP
Center	0.8950	0.9448	**0.8955**	0.8641
Donut	0.8514	0.8434	**0.7368**	0.7500
Edge-Loc	0.6721	0.7717	**0.7878**	0.7606
Edge-Ring	0.9628	0.9831	**0.9786**	0.9719
Loc	0.5838	0.6502	**0.7806**	0.7726
Near-full	0.9000	0.9286	**0.9310**	0.9167
None	0.6795	0.8624	**0.8655**	0.8571
Random	0.8963	0.9252	**0.8588**	0.7588
Scratch	0.7459	0.8601	**0.7802**	0.7011
Macro Average	0.7985	0.8633	**0.8461**	0.8170
Weighted Average	0.7861	0.8723	**0.8755**	0.8575

Table 4.3 Precision values for models trained on each dataset variation and pooling variation.

	Recall			
Class	RU	NR-NA	**NR-A-GAP**	NR-A-GMP
Center	0.7241	0.7579	**0.8580**	0.8661
Donut	0.5676	0.6306	**0.7568**	0.6486
Edge-Loc	0.5530	0.8468	**0.8584**	0.8324
Edge-Ring	0.9638	0.9618	**0.9680**	0.9654
Loc	0.4312	0.7135	**0.6036**	0.5341
Near-full	0.9000	0.8667	**0.9000**	0.7333
None	0.9422	0.9544	**0.9506**	0.9494
Random	0.8497	0.7861	**0.8786**	0.8728
Scratch	0.3824	0.5168	**0.5966**	0.5126
Macro Average	0.7015	0.7816	**0.8189**	0.7683
Weighted Average	0.7816	0.8659	**0.8766**	0.8599

Table 4.4 Recall values for models trained on each dataset variation and pooling variation.

	F1-Score			
Class	RU	NR-NA	**NR-A-GAP**	NR-A-GMP
Center	0.8005	0.8411	**0.8763**	0.8651
Donut	0.6811	0.7216	**0.7467**	0.6957
Edge-Loc	0.6068	0.8075	**0.8216**	0.7948
Edge-Ring	0.9633	0.9723	**0.9733**	0.9686
Loc	0.4960	0.6804	**0.6808**	0.6316
Near-full	0.9000	0.8966	**0.9153**	0.8148
None	0.7896	0.9061	**0.9060**	0.9009
Random	0.8724	0.8500	**0.8686**	0.8118
Scratch	0.5056	0.6457	**0.6762**	0.5922
Macro Average	0.7350	0.8135	**0.8294**	0.7862
Weighted Average	0.7726	0.8649	**0.8736**	0.8549

Table 4.5 F1-Score values for models trained on each dataset variation and pooling variation.

Across the classes, precision, recall and F1-Score values varied. For instance, the ‘Edge-Ring’ class consistently exhibited high precision and recall across all models, highlighting the class’s distinguishability. In contrast, the ‘Loc’ class demonstrated relatively lower precision and recall scores, further supporting our earlier observation that the models struggle to distinguish this class from other visually similar classes with different positions within the image. Among the configurations, the NR-A-GAP model achieve the highest weighted average precision and recall of 0.8755 and 0.8766 respectively. This shows that it is effective in reducing false positives while correctly identifying positive instances across diverse classes.

Figure 4.15, Figure 4.16, Figure 4.17, and Figure 4.18 shows the precision-recall curve for each model.

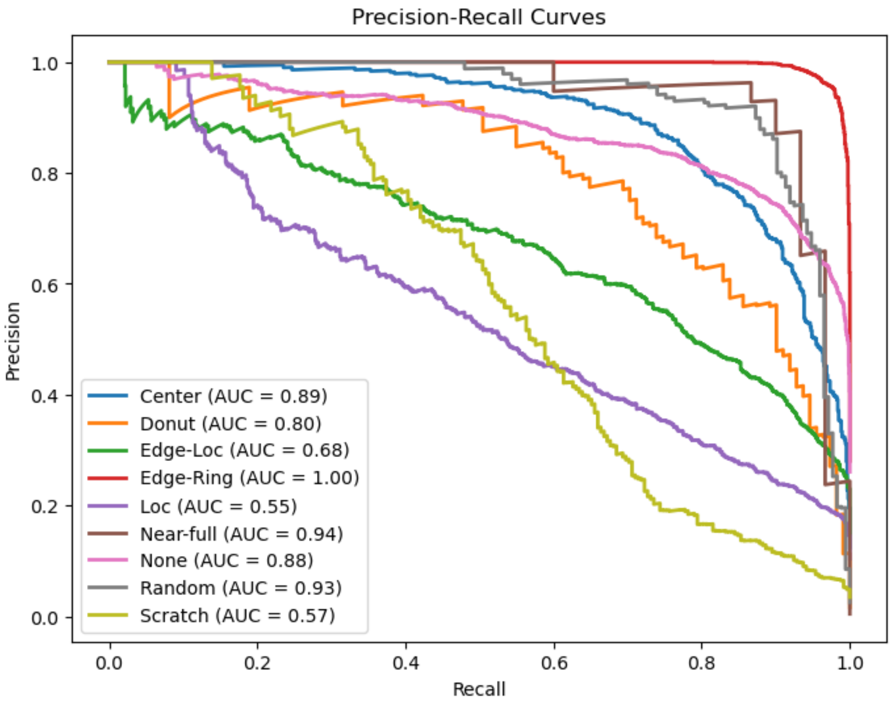


Figure 4.15 Precision-recall curve for RU model.

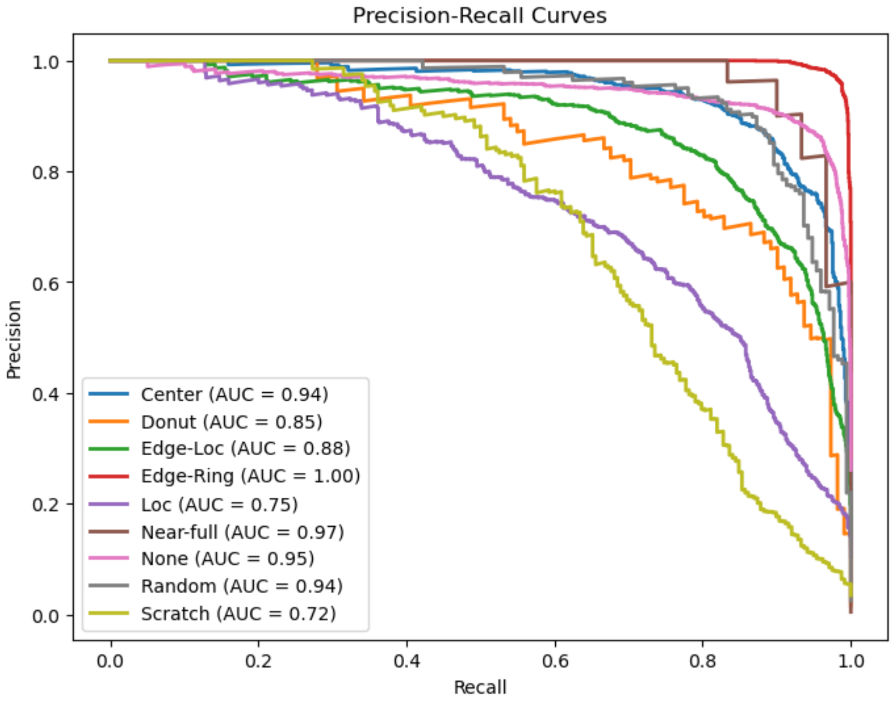


Figure 4.16 Precision-recall curve for NR-NA model.

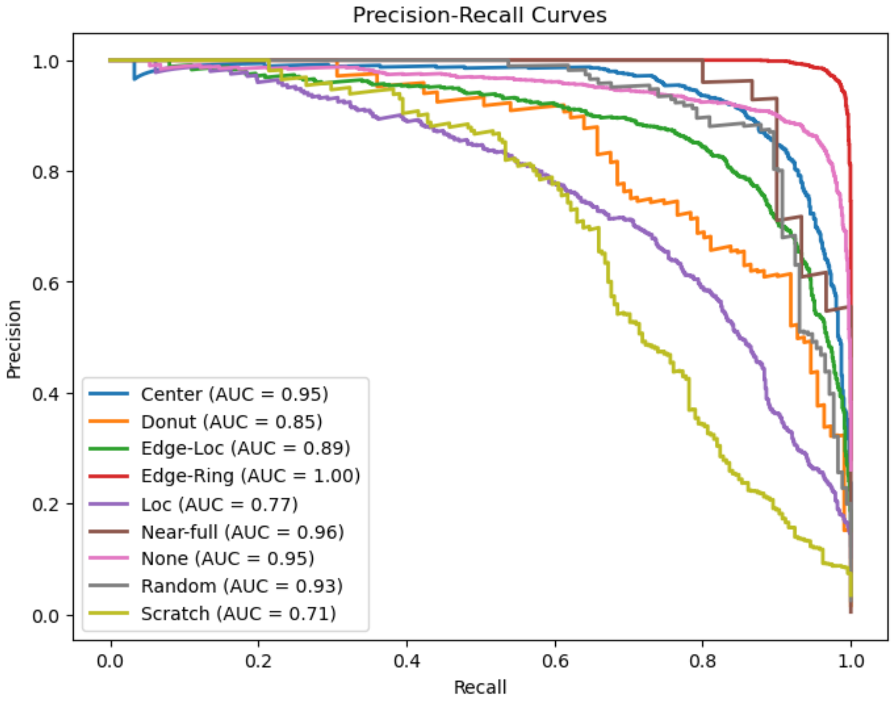


Figure 4.17 Precision-recall curve for NR-A-GAP model.

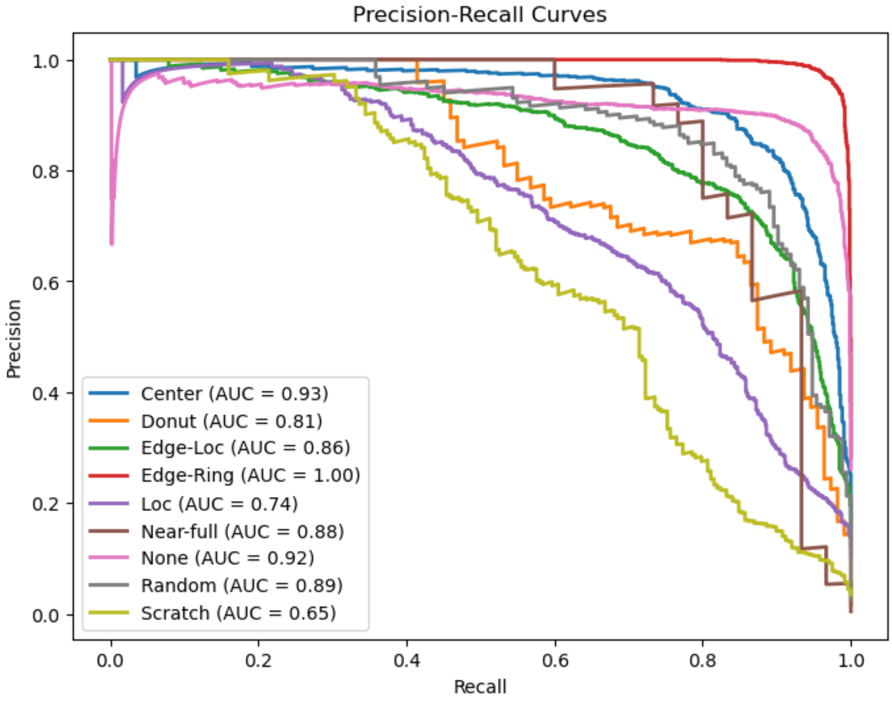


Figure 4.18 Precision-recall curve for NR-A-GMP model.

From the precision-recall curves, the use of noise reduction pre-processed data yields superior result. This is evident from the larger Area Under the Curve (AUC-PR) across all classes when comparing the NR-NA model to the RU model. Similar observations were made from the utilisation of augmented data and the implementation of GAP in the classification layer.

4.4 Hyperparameter Optimisation Phase Model Training and Evaluation

4.4.1 Training Result

Based on insights gathered from the data impact assessment phase, we train the models using different classification layer configurations as specified in Section 3.3.4 and fixed hyperparameters specified in Section 3.4.2 using the noise reduction pre-processed and augmented dataset..

Table 4.6 shows the training and validation accuracy and loss values at the end of the 40th epoch.

Config.	**Layer**	**Hidden Unit / Rate**	**Training**		**Validation**	
			Acc.	**Loss**	**Acc.**	**Loss**
A	Dense_layer_1	32	0.9058	0.2691	0.8949	0.2691
B	Dense_layer_1 Dense_layer_2 Dense_layer_3	128 64 32	0.9233	0.2203	0.9054	0.2886
C	Dense_layer_1 Dense_layer_2 Dropout	256 128 0.2	0.9191	0.2274	0.9038	0.2938

Table 4.6 Training and validation accuracy and loss values at the end of 40th epoch for each classification layer configuration.

At the end of training, the validation accuracy and loss for classification layer configuration B and C outperforms A. Configurations B and C, with their deeper layers, have a higher model capacity. This means the increased parameters can capture more complex patterns and representations from the wafer map data. While configuration C has a slightly lower accuracy and higher loss value at the end of training compared to configuration B, the introduction of the dropout layer prevents the model from overfitting, resulting in higher test accuracy, as demonstrated below.

In the second part of hyperparameter optimisation phase, we performed hyperparameter tuning on the model which yielded the most optimal result from the previous part. The hyperparameter settings for tuning are specified in Section 3.4.2.

Table 4.7 shows the training and validation accuracy and loss values at the end of training for each hyperparameter setting.

Hyperparameter	**Setting**	**Training**		**Validation**		**Computation time (s)**
		Acc.	**Loss**	**Acc.**	**Loss**	
Batch Size	16	0.9187	0.2293	0.9005	0.3056	5054
	32	**0.9191**	**0.2274**	**0.9038**	**0.2938**	**4540**
	64	0.9181	0.2306	0.9031	0.2912	4343
Learning Rate	0.01	0.8756	0.3629	0.8696	0.3945	
	0.001	**0.9191**	**0.2274**	**0.9038**	**0.2938**	
	0.0001	0.8861	0.3292	0.8825	0.3418	
Optimiser	**Adam**	**0.9191**	**0.2274**	**0.9038**	**0.2938**	
	RMSprop	0.8830	0.3824	0.8828	0.5131	
	SGD	0.7693	0.6798	0.7856	0.6406	

Table 4.7 Training and validation accuracy and loss values at the end of 40th epoch for different hyperparameter setting.

Among the tested batch sizes, 32 stands out as the optimal choice as it struck the right balance between model performance and training speed. For learning rate, 0.001 emerged as the most effective choice, exhibiting higher training and validation accuracy. On the contrary, 0.0001 showed suboptimal performance, possibly due to its excessively small steps during gradient descent, thus hindering the training process. When considering the optimisers, Adam outperformed RMSprop and SGD in terms of both training and validation accuracy. Adam’s adaptive learning rate and momentum mechanisms allowed for efficient convergence, contributing to its superior performance.

4.4.2 Model Evaluation

We then evaluate the models trained with the different hyperparameter settings on the test set. Table 4.8 shows the accuracy and loss values for each hyperparameter setting.

Hyperparameter	**Setting**	**Test**	
		Acc.	**Loss**
Batch Size	16	0.8863	0.4041
	32	**0.8917**	0.3800
	64	0.8870	0.3642
Learning Rate	0.01	0.8723	0.4441
	0.001	**0.8917**	0.3800
	0.0001	0.8828	0.3558
Optimiser	Adam	**0.8917**	0.3800
	RMSprop	0.8718	0.5493
	SGD	0.8090	0.6146

Table 4.8 Test accuracy and loss values for different hyperparameter settings.

When evaluated on the test set, the combination of a batch size of 32, a learning rate of 0.001, and the Adam optimiser produced superior results compared to the others.

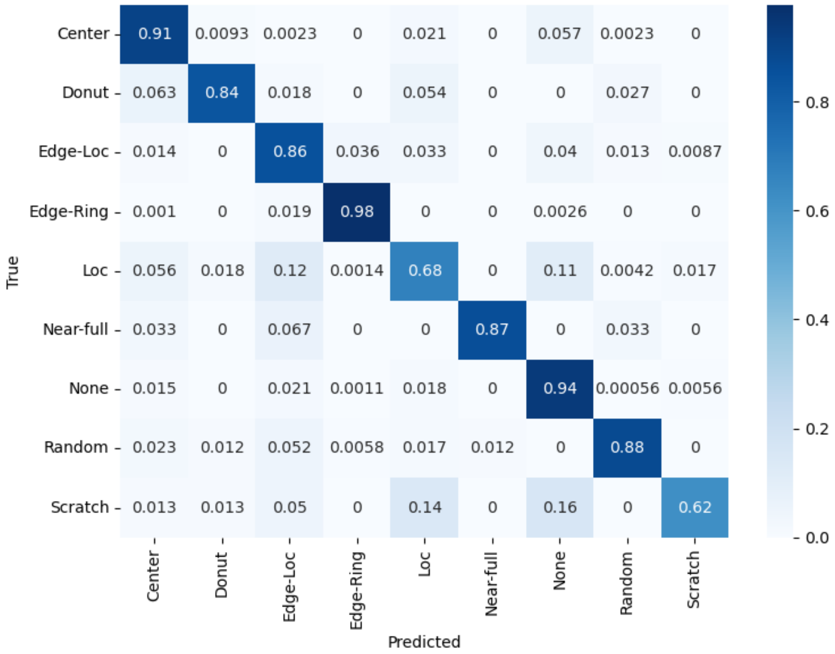
4.5 Finalised Model Evaluation and Analysis

Following hyperparameter tuning, we were able to obtain an optimal model that can accurately classify wafer map defect patterns. When evaluated on the test set, the model is able to achieve an accuracy of 89.17%. The hyperparameter settings for the finalised model are shown in.

Hyperparameter	**Value/Description**
Batch size	32
Initial learning rate	0.001
Optimiser	Adam
Classification layer configuration	Configuration B

Table 4.9 Hyperparameter settings for finalised model.

Figure 4.19 shows the normalised confusion matrix for the test set.



True \ Predicted	Center	Donut	Edge-Loc	Edge-Ring	Loc	Near-full	None	Random	Scratch
Center	0.91	0.0093	0.0023	0	0.021	0	0.057	0.0023	0
Donut	0.063	0.84	0.018	0	0.054	0	0	0.027	0
Edge-Loc	0.014	0	0.86	0.036	0.033	0	0.04	0.013	0.0087
Edge-Ring	0.001	0	0.019	0.98	0	0	0.0026	0	0
Loc	0.056	0.018	0.12	0.0014	0.68	0	0.11	0.0042	0.017
Near-full	0.033	0	0.067	0	0	0.87	0	0.033	0
None	0.015	0	0.021	0.0011	0.018	0	0.94	0.00056	0.0056
Random	0.023	0.012	0.052	0.0058	0.017	0.012	0	0.88	0
Scratch	0.013	0.013	0.05	0	0.14	0	0.16	0	0.62

Figure 4.19 Normalised confusion matrix for the test set evaluated using the optimal model.

Table 4.10 shows the precision, recall, and F1-Score for the test set.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Center	0.8874	0.9080	0.8976	859
Donut	0.7815	0.8378	0.8087	111
Edge-Loc	0.8260	0.8555	0.8405	1038
Edge-Ring	0.9788	0.9778	0.9783	1936
Loc	0.7928	0.6759	0.7297	719
Near-full	0.9286	0.8667	0.8966	30
None	0.8885	0.9389	0.9130	1800
Random	0.8686	0.8786	0.8736	173
Scratch	0.8268	0.6218	0.7098	238

Table 4.10 Precision, recall and F1-Score values for the test set evaluated using the optimal model.

Figure 4.20 shows the precision-recall curve for the test set.

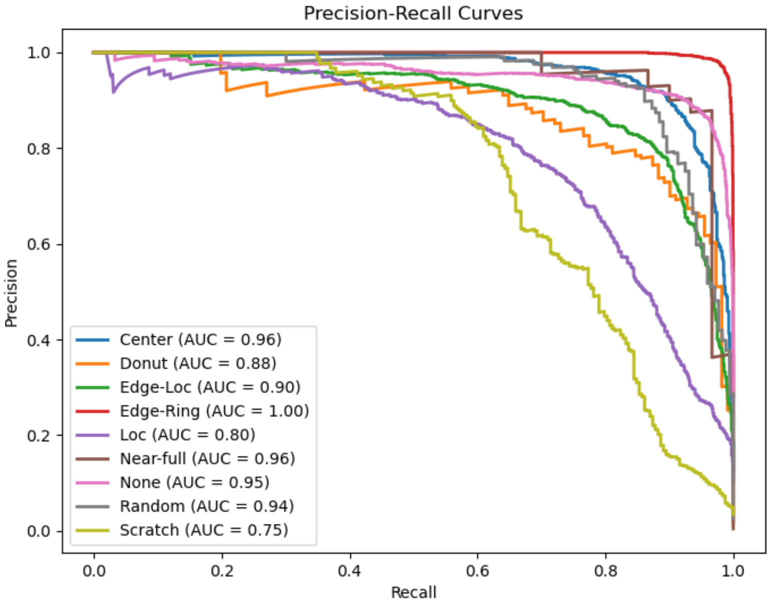


Figure 4.20 Precision-recall curve for the test set evaluated using the optimal model.

From the above metrics, it can be observed that the optimal model is able to correctly classify most of the classes within the test set, with the exception of the 'Loc' and 'Scratch' classes. As with other models evaluated in this research, the spatial discrimination limitation between these two classes poses significant challenges to CNNs due to their similar visual features but differ in their spatial placements on the wafer map. Instances where the model misclassified 'Loc', 'Edge-Loc', and 'Center' due to their similarity is shown in Figure 4.21.

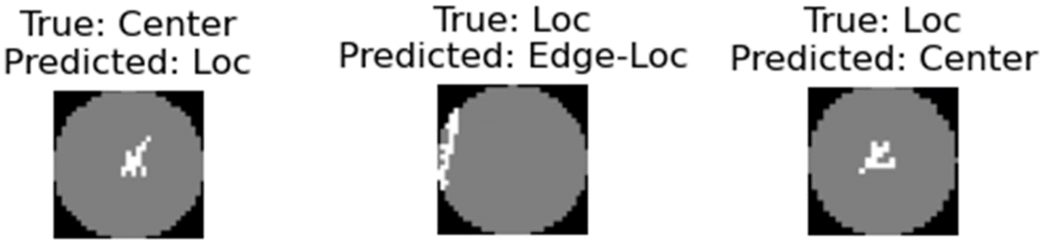


Figure 4.21 Misclassification due to different spatial localisation in similar classes.

Table 4.11 and Table 4.12 shows the comparison of data handling methods and test accuracy between previous methods and our proposed method.

Ref.	Methods	Dataset	Classes	Data Aug.
[19]	Custom-made CNN	Real and Synthetic	22	None
[20]	Transfer learning using CNN	WM-811K	9	None
-	**Proposed method**	**WM-811K**	**9**	**Data Generation using CAE**

Table 4.11 Comparison of dataset handling between previous works and our proposed method.

Ref.	Methods	Imbalance Control	Sample Size		Test Acc.
			Train	Test	
[19]	Custom-made CNN	None	28600	6600	98.20%
[20]	Transfer learning using CNN	Downsampling	7112	2000	96.80%
-	**Proposed method**	**Upsampling using synthetic data generated from CAE**	**69278**	**6904**	**89.17%**

Table 4.12 Comparison of training sample size, test sample size and accuracy between previous works and our proposed method.

While our proposed method may not achieve the same high accuracy as the previous works, it is important to highlight a key differentiator in our approach. Our research places a strong emphasis on recognising and accommodating the diverse internal variations within each class of wafer map defect patterns. Hence, we have chosen to include the whole labelled WM-811K to train our model and generate additional synthetic data to upsample the minority classes to address the class imbalance issue in this dataset. This focus on capturing the intricate nuances within each class allowed us to gain a more robust understanding of the defect pattern which aided in the design of our model.

Furthermore, it is worth nothing that the framework proposed by [19] was to train CNN model primarily on synthetic data. While the use of synthetic data can be effective in some cases, it may introduce challenges to the model's performance on real-world data. Our approach leverages a combination of real and synthetic data generated through CAE to strike a balance between the richness of real-world data and the controlled variability of synthetic data. To address the class imbalance issue, [20] chose to downsample their dataset. While this can mitigate class imbalance, it may unintentionally discard certain internal variations within a class. This may affect the model's performance when dealing with rare and varied defect patterns in real-world scenarios.

Therefore, while our test accuracy may not be the highest, our research strives to address the complexities of real-world wafer map data by considering internal variations within each class. By avoiding downsampling and integrating synthetic data, we aim to preserve the richness of the data, allowing our model to better adapt to the diverse defect patterns encountered in practical applications.

4.6 Implementation Issues and Challenges

While conducting this research, we encountered several noteworthy challenges, each of which contributed valuable insights to our study. They are as follows:

4.6.1 Poor Spatial Discrimination between Similar Classes

One of the prominent challenges we confronted was the issue of poor spatial discrimination between classes that exhibit visual similarities but crucially differ in their spatial characteristics. These classes, despite their visual similarities, necessitate precise spatial awareness for accurate classification.

The challenge arises from the nature of CNNs, which excel in recognising patterns and features but inherently possess a limitation in handling spatial context. CNNs are fundamentally designed to be translation invariant, where the number of convolutional layers, number of pooling layers, filter size, and data augmentation plays a role in achieving this invariance [29], [30]. This property is advantageous in numerous classification scenarios; however, it becomes a hurdle when class discrimination relies heavily on spatial cues.

4.6.2 Limitations of the Dataset

Owing to the wide range of variations present within individual classes of wafer maps, some of which closely resemble variations in other classes, our model faces challenges in effectively distinguishing between defect patterns during classification. For example, Figure 4.22 shows a ‘Loc’ defect pattern and a ‘Scratch’ defect pattern which have similar features.

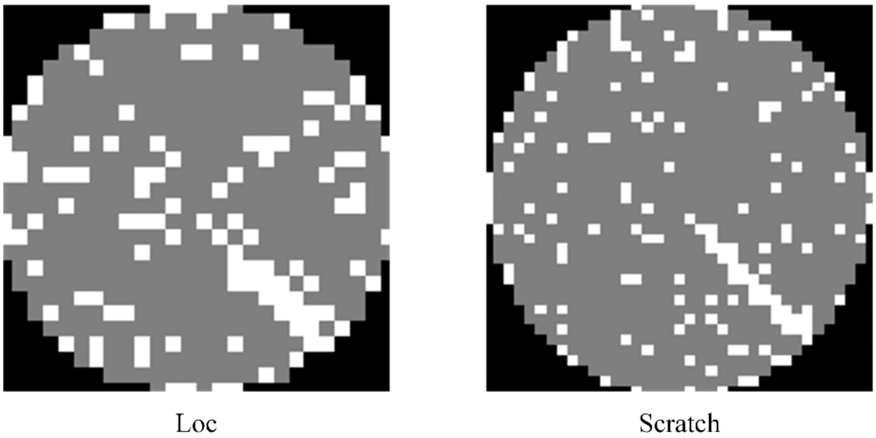


Figure 4.22 Similar features on two different classes before noise reduction.

The removal of noise, while intended to enhance data quality, may have inadvertently exacerbated the challenge in certain cases, as evident in Figure 4.23.

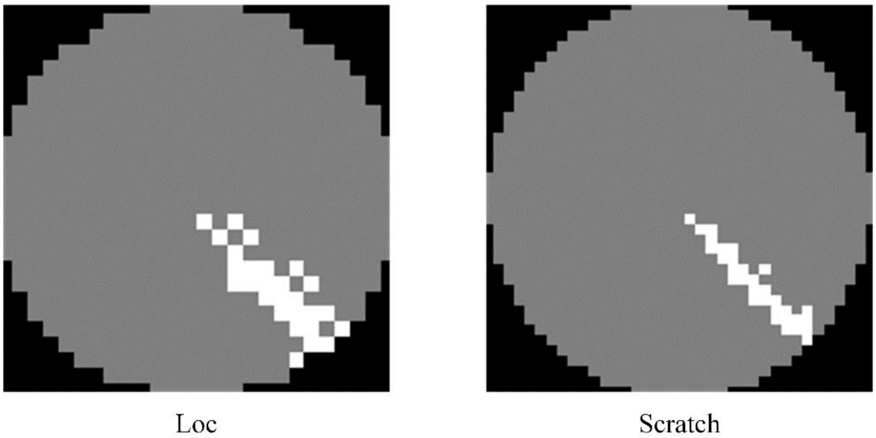


Figure 4.23 Similar features on two different classes after noise reduction.

4.6.3 Limitations of Synthetic Wafer Maps Generated by CAE

The images generated by the CAE might have lacked sufficient diversity, as they closely resembled the original images. This could have limited the model’s ability to learn from a more varied set of examples and impacted its overall performance.

CHAPTER 5 CONCLUSION

In the past, traditional wafer inspections relied on manual examination by human experts. This approach could result in inaccurate classifications due to fatigue and bias, as the experts had to inspect a substantial number of wafers. The existing literature reveals that various deep learning techniques have been investigated to classify wafer defect patterns in the past. Nonetheless, the requirement for large datasets to achieve optimal performance with deep learning models, coupled with the scarcity of extensive and high-quality wafer map datasets, and the presence of relatively rare defect patterns resulting in imbalanced dataset, may hinder the effectiveness of custom-made CNNs for defect pattern classification. To address this limitation, a novel approach involving the use of CCL for noise reduction, CAE for data augmentation, and transfer learning method using EfficientNet model was proposed to create a robust deep learning model that can effectively classify wafer map defect patterns.

Prior to training the model, data pre-processing has been carried out by highlighting the defect pattern clusters using CCL with 8-connected neighbourhood and reducing the random noise in the wafer maps. The imbalanced wafer map dataset is then augmented using CAE to generate synthetic wafer map data. After training and extensive hyperparameter tuning, an optimal model was obtained which achieved an accuracy of 89.17% when evaluated on the test set. Our model also achieved a weighted average F1-Score of 0.8897 which shows that it is not only accurate in classifying wafer map defect patterns but is also capable to distinguish different defect patterns in imbalanced dataset, which are often the case in real-world scenarios. However, the misclassification of classes, particularly the ‘Loc’ and ‘Edge-Loc’ classes indicates that the classification framework can be further improved.

There are several insights and limitations observed throughout the course of this research which would benefit from further research. Due to the wide range of variations within some classes such as ‘Loc’, a dataset that separates the variations into different subclasses may be beneficial compared to the current WM-811K. Future research may also consider the use of transformer networks or incorporates attention mechanisms into the CNN architecture to focus on relevant spatial regions in the wafer map image.

REFERENCES

[1] S. K. Saha, ‘Modeling process variability in scaled CMOS technology’, *IEEE Design & Test of Computers*, vol. 27, no. 2, pp. 8–16, 2010, doi: 10.1109/MDT.2010.50.

[2] K. J. Kuhn *et al.*, ‘Process technology variation’, *IEEE Trans Electron Devices*, vol. 58, no. 8, pp. 2197–2208, 2011, doi: 10.1109/TED.2011.2121913.

[3] C. K. Hansen and P. Thyregod, ‘Use of wafer maps in integrated circuit manufacturing’, *Microelectronics Reliability*, vol. 38, no. 6, pp. 1155–1164, 1998, doi: https://doi.org/10.1016/S0026-2714(98)00127-9.

[4] N. G. Shankar and Z. W. Zhong, ‘Defect detection on semiconductor wafer surfaces’, *Microelectron Eng*, vol. 77, no. 3, pp. 337–346, 2005, doi: https://doi.org/10.1016/j.mee.2004.12.003.

[5] J. S. Pettinato and D. Pillai, ‘Technology decisions to minimize 450-mm wafer size transition risk’, *IEEE Transactions on Semiconductor Manufacturing*, vol. 18, no. 4, pp. 501–509, 2005, doi: 10.1109/TSM.2005.858471.

[6] S. P. Cunningham and S. MacKinnon, ‘Statistical methods for visual defect metrology’, *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 1, pp. 48–53, 1998, doi: 10.1109/66.661284.

[7] W. Taam and M. Hamada, ‘Detecting spatial effects from factorial experiments: An application from integrated-circuit manufacturing’, *Technometrics*, vol. 35, no. 2, pp. 149–160, Aug. 1993, doi: 10.2307/1269660.

[8] L. Xie, R. Huang, and Z. Cao, ‘Detection and classification of defect patterns in optical inspection using support vector machines’, in *Intelligent Computing Theories*, D.-S. Huang, V. Bevilacqua, J. C. Figueroa, and P. Premaratne, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 376–384.

[9] M.-J. Wu, J.-S. R. Jang, and J.-L. Chen, ‘Wafer map failure pattern recognition and similarity ranking for large-scale data sets’, *IEEE Transactions on Semiconductor Manufacturing*, vol. 28, no. 1, pp. 1–12, 2015, doi: 10.1109/TSM.2014.2364237.

[10] B. Kim, Y.-S. Jeong, S. H. Tong, I.-K. Chang, and M.-K. Jeongyoung, ‘A regularized singular value decomposition-based approach for failure pattern classification on fail bit map in a DRAM wafer’, *IEEE Transactions on Semiconductor Manufacturing*, vol. 28, no. 1, pp. 41–49, 2015, doi: 10.1109/TSM.2014.2388192.

[11] U. Batool, M. I. Shapiai, M. Tahir, Z. H. Ismail, N. J. Zakaria, and A. Elfakharany, ‘A systematic review of deep learning for silicon wafer defect recognition’, *IEEE Access*, vol. 9, pp. 116572–116593, 2021, doi: 10.1109/ACCESS.2021.3106171.

[12] S. Albawi, T. A. Mohammed, and S. Al-Zawi, ‘Understanding of a convolutional neural network’, in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. doi: 10.1109/ICEngTechnol.2017.8308186.

[13] S. J. Pan and Q. Yang, ‘A survey on transfer learning’, *IEEE Trans Knowl Data Eng*, vol. 22, no. 10, pp. 1345–1359, 2010, doi: 10.1109/TKDE.2009.191.

[14] G. Huang, Z. Liu, and K. Q. Weinberger, ‘Densely connected convolutional networks’, *CoRR*, vol. abs/1608.0, 2016, [Online]. Available: http://arxiv.org/abs/1608.06993

[15] K. He, X. Zhang, S. Ren, and J. Sun, ‘Deep residual learning for image recognition’, *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[16] M. Tan and Q. V Le, ‘EfficientNet: Rethinking model scaling for convolutional neural networks’, in *36th International Conference on Machine Learning, ICML 2019*, 2019, pp. 10691–10700.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ‘ImageNet: A large-scale hierarchical image database’, in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[18] F.-L. Chen and S.-F. Liu, ‘A neural-network approach to recognize defect spatial pattern in semiconductor fabrication’, *IEEE Transactions on*

Semiconductor Manufacturing, vol. 13, no. 3, pp. 366–373, 2000, doi: 10.1109/66.857947.

[19] T. Nakazawa and D. V Kulkarni, ‘Wafer map defect pattern classification and image retrieval using convolutional neural network’, *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 2, pp. 309–314, 2018, doi: 10.1109/TSM.2018.2795466.

[20] J. Yu, Z. Shen, and S. Wang, ‘Wafer map defect recognition based on deep transfer learning-based densely connected convolutional network and deep forest’, *Eng Appl Artif Intell*, vol. 105, p. 104387, 2021, doi: https://doi.org/10.1016/j.engappai.2021.104387.

[21] W. Liu and K. Zeng, ‘SparseNet: {A} Sparse DenseNet for image classification’, *CoRR*, vol. abs/1804.0, 2018, [Online]. Available: http://arxiv.org/abs/1804.05340

[22] M. Grandini, E. Bagli, and G. Visani, ‘Metrics for Multi-Class Classification: An Overview’, Aug. 2020.

[23] H. He and E. A. Garcia, ‘Learning from Imbalanced Data’, *IEEE Trans Knowl Data Eng*, vol. 21, no. 9, pp. 1263–1284, 2009, doi: 10.1109/TKDE.2008.239.

[24] M. B. Alawieh, D. Boning, and D. Z. Pan, ‘Wafer Map Defect Patterns Classification using Deep Selective Learning’, in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6. doi: 10.1109/DAC18072.2020.9218580.

[25] J. Cha, S. Oh, D. Kim, and J. Jeong, ‘A Defect Detection Model for Imbalanced Wafer Image Data Using CAE and Xception’, in *2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, 2020, pp. 28–33. doi: 10.1109/IDSTA50958.2020.9264135.

[26] M. Lin, Q. Chen, and S. Yan, ‘Network In Network’, *CoRR*, vol. abs/1312.4400, 2013.

[27] E. Maeland, ‘On the comparison of interpolation methods’, *IEEE Trans Med Imaging*, vol. 7, no. 3, pp. 213–217, 1988, doi: 10.1109/42.7784.

[28] D. Han, ‘Comparison of Commonly Used Image Interpolation Methods’, in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013),* Atlantis Press, Mar. 2013, pp. 1556–1559. doi: 10.2991/iccsee.2013.391.

[29] H. Furukawa, ‘Deep Learning for Target Classification from SAR Imagery: Data Augmentation and Translation Invariance’, Aug. 2017.

[30] E. Kauderer-Abrams, ‘Quantifying Translation-Invariance in Convolutional Neural Networks’, Dec. 2017.

WEEKLY LOG

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	**Study week no.: 2**
Student Name & ID: Lim Yu Pin 1906069	
Supervisor: Dr Lim Jia Qi	
Project Title: Wafer Map Defect Pattern Classification using Deep Learning Model	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Review FYP1 work and decide the direction of FYP2.
- Add new information to be taken into consideration for FYP2.

2. WORK TO BE DONE

- Read up and update on literature review since the end of FYP1.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

- I need to catch up to speed on what has been done.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	**Study week no.: 4**
Student Name & ID: Lim Yu Pin 1906069	
Supervisor: Dr Lim Jia Qi	
Project Title: Wafer Map Defect Pattern Classification using Deep Learning Model	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Tried semi-supervised learning using pseudo-labelling method for classification after discussion with supervisor.
- Pseudo-labelling method unable to obtain significant improvement compared to previous methods.

2. WORK TO BE DONE

- Explore more ways to address the class imbalance issue.

3. PROBLEMS ENCOUNTERED

- Ran into RAM limits. Solve by using Google Colab temporarily.

4. SELF EVALUATION OF THE PROGRESS

- Good progress.
- Good discussion with supervisor and gain insights on direction of research.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	**Study week no.: 6**
Student Name & ID: Lim Yu Pin 1906069	
Supervisor: Dr Lim Jia Qi	
Project Title: Wafer Map Defect Pattern Classification using Deep Learning Model	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Tried semi-supervised GAN model to generate additional synthetic data.

2. WORK TO BE DONE

- Review overall pipeline, including data splitting and data pre-processing steps to ensure it is correct.

3. PROBLEMS ENCOUNTERED

- Semi-supervised GAN or pseudo-labelling model would not be able to work well due to the nature of the dataset.
- Not only is the labelled set imbalanced, from observation, the unlabeled set is also imbalanced and dominated by a few classes.

4. SELF EVALUATION OF THE PROGRESS

- Slight setback to the progress as we found that the semi-supervised model would not work for our case.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	**Study week no.: 8**
Student Name & ID: Lim Yu Pin 1906069	
Supervisor: Dr Lim Jia Qi	
Project Title: Wafer Map Defect Pattern Classification using Deep Learning Model	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Revert to model and framework used in FYP1.

2. WORK TO BE DONE

- Set out the flow of the model training and evaluation.
- Retrain CAE on wafer maps data using the subscribed Google Colab Pro to generate higher resolution and quality synthetic wafer maps.

3. PROBLEMS ENCOUNTERED

None

4. SELF EVALUATION OF THE PROGRESS

- Have to pick up the pace due to slight setback.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	**Study week no.: 10**
Student Name & ID: Lim Yu Pin 1906069	
Supervisor: Dr Lim Jia Qi	
Project Title: Wafer Map Defect Pattern Classification using Deep Learning Model	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Performed training and evaluation for baseline model.
- Investigated the impact of different types of data (ie. Non-augmented, augmented) on the performance of the model.

2. WORK TO BE DONE

- Outline all hyperparameter to be considered in the hyperparameter optimisation phase.
- Train the model using data with noise removed.
- Start to compile report.

3. PROBLEMS ENCOUNTERED

- Model performance, especially accuracy is not as robust as expected. However, this brought to our attention the underlying nature of the dataset.
- However, with the model trained on data with noise reduced, we were able to increase the model's performance slightly.

4. SELF EVALUATION OF THE PROGRESS

- Good progress, allocated tasks are completed as scheduled.

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Trimester 3, Year 3	**Study week no.: 12**
Student Name & ID: Lim Yu Pin 1906069	
Supervisor: Dr Lim Jia Qi	
Project Title: Wafer Map Defect Pattern Classification using Deep Learning Model	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Performed hyperparameter tuning on the model.
- Analysis of the model's performance.

2. WORK TO BE DONE

- Finalise and complete remaining report for FYP2.

3. PROBLEMS ENCOUNTERED

- Slightly behind schedule on report writing as we tried to boost the performance of the model the previous week.

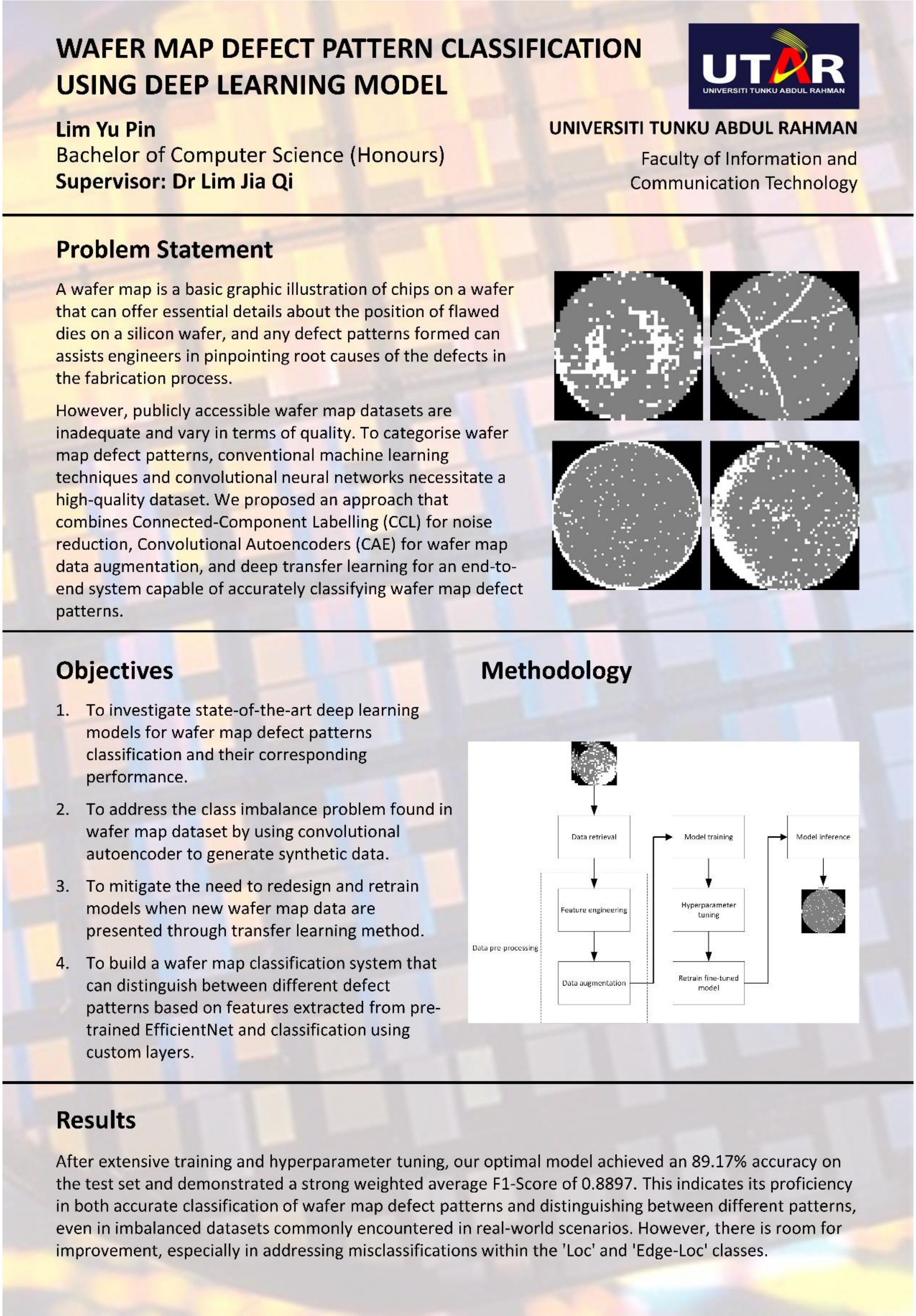
4. SELF EVALUATION OF THE PROGRESS

- Gathered valuable insights and knowledge regarding the domain that I researched.

Supervisor's signature

Student's signature

POSTER



WAFER MAP DEFECT PATTERN CLASSIFICATION USING DEEP LEARNING MODEL

Lim Yu Pin
Bachelor of Computer Science (Honours)
Supervisor: Dr Lim Jia Qi

UNIVERSITI TUNKU ABDUL RAHMAN
Faculty of Information and
Communication Technology

Problem Statement

A wafer map is a basic graphic illustration of chips on a wafer that can offer essential details about the position of flawed dies on a silicon wafer, and any defect patterns formed can assists engineers in pinpointing root causes of the defects in the fabrication process.

However, publicly accessible wafer map datasets are inadequate and vary in terms of quality. To categorise wafer map defect patterns, conventional machine learning techniques and convolutional neural networks necessitate a high-quality dataset. We proposed an approach that combines Connected-Component Labelling (CCL) for noise reduction, Convolutional Autoencoders (CAE) for wafer map data augmentation, and deep transfer learning for an end-to-end system capable of accurately classifying wafer map defect patterns.

Objectives

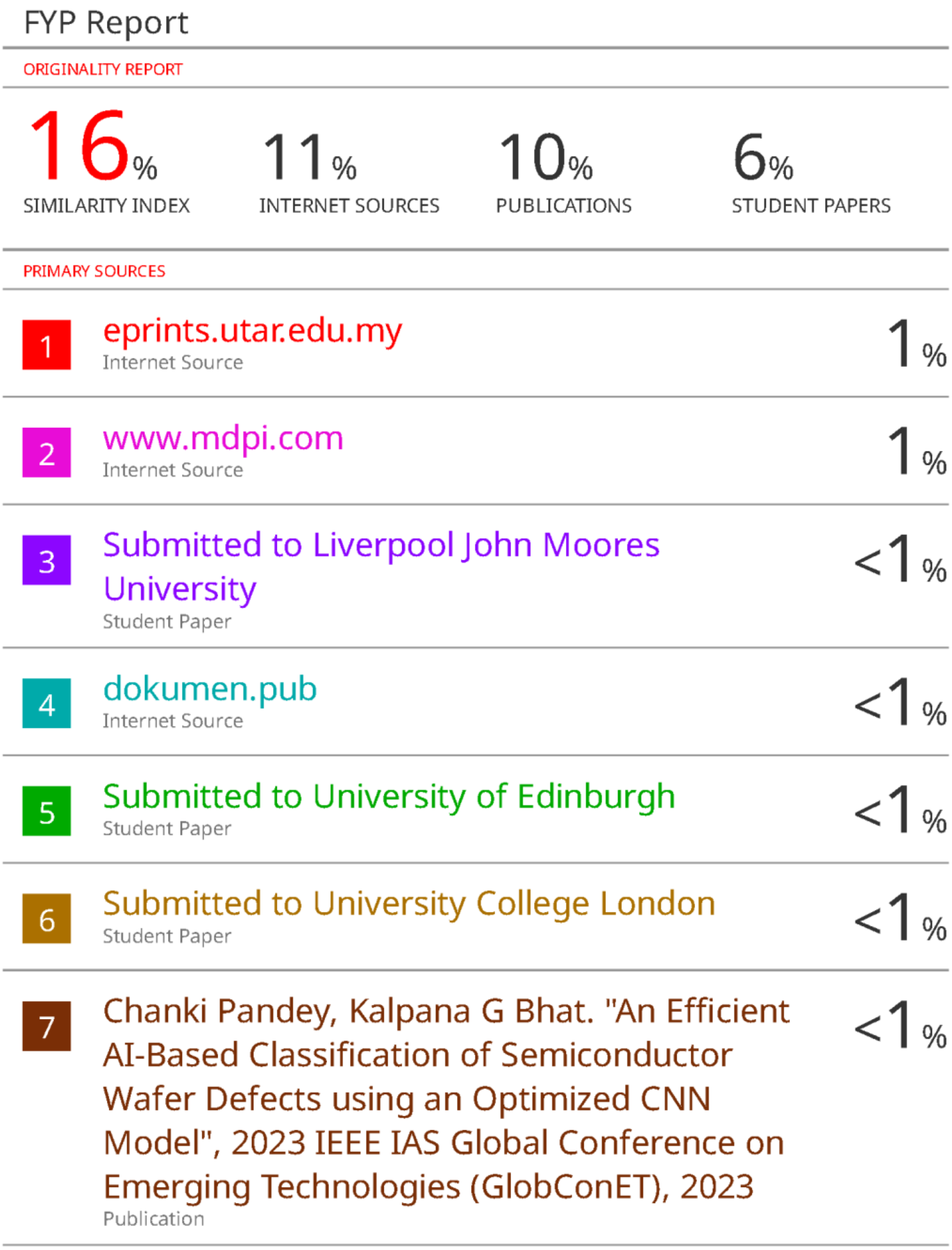
1. To investigate state-of-the-art deep learning models for wafer map defect patterns classification and their corresponding performance.
2. To address the class imbalance problem found in wafer map dataset by using convolutional autoencoder to generate synthetic data.
3. To mitigate the need to redesign and retrain models when new wafer map data are presented through transfer learning method.
4. To build a wafer map classification system that can distinguish between different defect patterns based on features extracted from pre-trained EfficientNet and classification using custom layers.

Methodology

Results

After extensive training and hyperparameter tuning, our optimal model achieved an 89.17% accuracy on the test set and demonstrated a strong weighted average F1-Score of 0.8897. This indicates its proficiency in both accurate classification of wafer map defect patterns and distinguishing between different patterns, even in imbalanced datasets commonly encountered in real-world scenarios. However, there is room for improvement, especially in addressing misclassifications within the 'Loc' and 'Edge-Loc' classes.

PLAGIARISM CHECK RESULT



FYP Report

ORIGINALITY REPORT

16% SIMILARITY INDEX

11% INTERNET SOURCES

10% PUBLICATIONS

6% STUDENT PAPERS

PRIMARY SOURCES

1	eprints.utar.edu.my Internet Source	1%
2	www.mdpi.com Internet Source	1%
3	Submitted to Liverpool John Moores University Student Paper	<1%
4	dokumen.pub Internet Source	<1%
5	Submitted to University of Edinburgh Student Paper	<1%
6	Submitted to University College London Student Paper	<1%
7	Chanki Pandey, Kalpana G Bhat. "An Efficient AI-Based Classification of Semiconductor Wafer Defects using an Optimized CNN Model", 2023 IEEE IAS Global Conference on Emerging Technologies (GlobConET), 2023 Publication	<1%

PLAGIARISM CHECK RESULT

Universiti Tunku Abdul Rahman			
Form Title: Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Lim Yu Pin
ID Number(s)	19ACB06069
Programme / Course	Bachelor of Computer Science (Honours)
Title of Final Year Project	Wafer Map Defect Pattern Classification using Deep Learning Model

Similarity	**Supervisor's Comments** **(Compulsory if parameters of originality exceed the limits approved by UTAR)**
Overall similarity index: 16% **Similarity by source** Internet Sources: 11% Publications: 10% Student Papers: 6%	
Number of individual sources listed of more than 3% similarity: 0	

Parameters of originality required, and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***

Note: Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Signature of Supervisor	Signature of Co-Supervisor
Name: Lim Jia Qi	Name:
Date: 14/09/2023	Date:

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	1906069
Student Name	LIM YU PIN
Supervisor Name	DR. LIM JIA QI

TICK (√)	**DOCUMENT ITEMS** Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.

(Signature of Student)
Date: 12/09/2023