Journal of Theoretical and Applied Information Technology
15th February 2024. Vol.102. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

DEEP NEURAL NETWORKS FOR EFFICIENT CLASSIFICATION OF SINGLE AND MIXED DEFECT PATTERNS IN SILICON WAFER MANUFACTURING

HUSSEIN YOUNIS[1], AMJAD RATTROUT[2, *], MOHAMMED YOUNIS[3]

[12]Arab American University, Department of Natural, Engineering and Technology Sciences, Palestine

[3]University of Memphis, Department of Electrical Engineering and Computer Engineering, United States

*Corresponding Author

E-mail: [1]Hussein.Younis@aaup.edu, [2]amjad.rattrout@aaup.edu, [3]myounis@memphis.edu

ABSTRACT

Semiconductor wafer manufacturing is a complex and costly process with inherent defect risks that can significantly impact the industry. Utilizing Deep Learning (DL) for wafer defect classification offers benefits such as improved performance, reduced human error, and time savings. This paper presents an advanced DL-based approach for wafer defect classification, based on a modified GoogLeNet model and data augmentation technique. The approach achieves state-of-the-art results on the WM-300K+ wafer map dataset, demonstrating robustness to image noise and variation. Our research introduces a pioneering approach that outperforms previous methodologies, integrating auto-cast and CUDA to enhance efficiency, addressing dataset imbalance through innovative data augmentation, and creating a new "WM-300K+ wafer map [Single & Mixed]" dataset. The methodology yields exceptional results, with an average classification accuracy of 99.9% for both single and mixed defect types, surpassing previous studies. Hyperparameter tuning with Optuna and a patient stop mechanism further fortifies the robustness and reliability of the approach.

Keywords: *Semiconductor Wafer Manufacturing, Deep Learning, Convolutional Neural Network, Wafer Defect Patterns, GoogLeNet*

1. INTRODUCTION

The demand for electronic devices has significantly increased owing to the Fourth Industrial Revolution (Industry 4.0), advancements in semiconductor manufacturing, and Internet of Things (IoT) devices [1], [2]. According to Fortune Business Insights statistics, the global consumer electronics market is projected to reach 989.37 billion USD by 2027 [3]. Electronic devices are composed of integrated circuits that contain various electronic components, including resistors, transistors, and diodes. These components and their connections are built on a semiconductor wafer typically composed of single-crystal silicon (Si). A wafer is a thin, circular slice of material that serves as the foundation for the production of integrated circuits. The process of fabricating integrated circuits involves transforming raw materials or components into a final product. Semiconductor wafer manufacturing, which is at the core of integrated circuit production, is a highly complex process that involves numerous steps and requires advanced equipment and expertise [4].

Semiconductor wafer manufacturing involves complex processes such as wafer processing, oxidation, photomask, etching, film deposition, interconnection, testing, and packaging [5]. Throughout these processes, the occurrence of defects is significant. Semiconductor engineers rely on complex and time-consuming diagnosis, inspection, and analysis processes in each phase to detect defects. These processes help track and address the source of failure before reaching the final production stage [6]. They aim to ensure that components are aligned correctly and operate as intended. Defects can occur due to dust particles and human errors [7]. Engineers use diagnostic methods to detect defects, which typically take up to 26 weeks [8]. Training operators or engineers to classify defects manually with 90% accuracy can take up to 9 months [9].

Wafer defect patterns play a crucial role in identifying specific manufacturing issues. These

patterns are spatially distributed from the edge to the center of the wafer surface and provide valuable information. The most common defect patterns include center, donut, edge-loc, edge-ring, loc, near-full, random, and scratch [10]. **Figure 1** illustrates the visualization of these common defect patterns in the wafer maps.

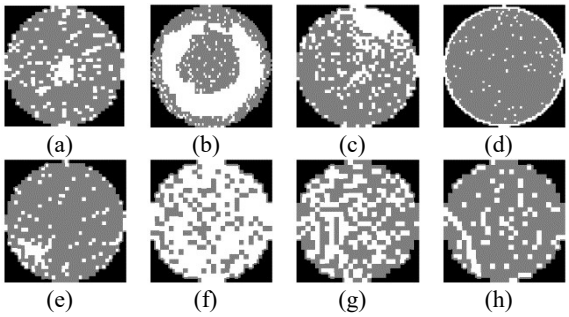


Figure 1: Visualization of common defect patterns in wafer maps, including (a) center, (b) donut, (c) edge-loc, (d) edge-ring, (e) loc, (f) near-full, (g) random and (h) scratch.

The wafer's defects may stem from various issues in the manufacturing process. For instance, mishandling by the machine often causes scratches, whereas problems during etching are typically linked to edge-ring defects. Center defects can indicate complications arising from thin-film deposition techniques [11]. The automated inspection machine tests that produce the wafer maps are shown in Figure 2. Wafer maps are visualizations based on abnormal locations on silicon wafers and other important information for tracking and manufacturing processes [12]. The wafer map provides insight into whether each wafer die meets the performance standards. By analyzing the spatial pattern of wafer maps, engineers can determine whether production meets standards and identify wafers that contain defects [13]. Although manual defect detection is not ideal, human experts achieve less than 45% accuracy, and it is time-consuming, costly, complicated, and highly disciplined [14].

In recent years, deep learning (DL), a subset of neural networks capable of handling unstructured data such as images and audio, has gained prominence [15]. DL can automatically extract features from data without human intervention, making it well-suited for wafer defect classification. By leveraging DL, performance can be improved while reducing human error and time. However, one limitation in this field is the limited availability of public datasets and access restrictions to real-world data for defect classification on wafers. These limitations hinder the generalizability and applicability of the research findings in this area. The integration of DL into wafer defect classification has been extensively researched and developed. Many studies have focused on using wafer map images and implementing various pre-processing and learning strategies to effectively learn wafer map defect patterns [16].

The Convolutional Neural Network (CNN), is a class of DL models that was first invented by Yann LeCun and others in 1998. CNNs are commonly used in supervised learning tasks, aiming to learn the mapping between input data and corresponding output labels [17]. They are particularly effective in handling array-like data such as RGB images, automatically extracting relevant information through their layers. The term "convolutional" in CNN refers to the utilization of convolutional layers, which extract spatial features from images. This makes CNNs more efficient and effective for image-related tasks compared to Artificial Neural Networks (ANNs) [18]. Typically, a CNN consists of an input layer, hidden layers, and an output layer [18]. The hidden layers include one or more layers that perform convolutions, as depicted in Figure 2.

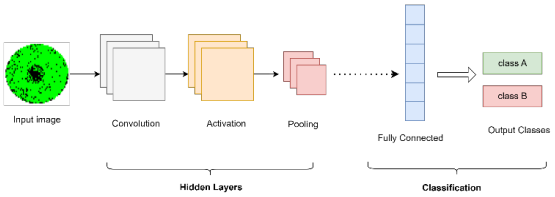


Figure 2: The Architecture of CNN.

During the feedforward propagation phase in CNN, the input data are passed through different layers, including convolutional, pooling, and fully connected layers. Activation functions and weight parameters are applied at each layer to transform the input and generate the output. The output of one layer serves as the input to the next layer, and this process continues until the final output is obtained. However, random initialization of weights and biases during the feedforward propagation phase can lead to errors in achieving the desired output. To address this issue, a mechanism called backpropagation is employed. Backpropagation involves calculating the gradients of the loss function concerning the weights and biases in the network. These gradients are then used to iteratively update the weights and biases, optimizing the network's performance and reducing the errors during training.

This study presents a wafer defect classification model that uses deep learning techniques,

specifically by adapting a deep CNN. The model takes wafer-map images as inputs and employs data preprocessing and data augmentation operations to enhance the classification of single- or mixed-type wafer map defects. The motivation behind this study was to develop an accurate and efficient system for classifying silicon wafer defects by leveraging DL techniques. However, DL techniques show promise for accurate and automated defect classification, major challenge is the limited availability of public datasets specifically designed for wafer map defects. The existing public 'WM-811k' dataset suffers from class imbalance issues. A new dataset called "Mixed-type Wafer Defect" has been derived from the 'WM-811k' dataset to overcome class imbalance issues. This new dataset focuses on capturing mixed types of wafer-map defect patterns, enabling researchers to develop more robust and accurate deep-learning models for defect classification. In addition, researchers face access restrictions to electronic wafer maps and integrated circuit designs as they are proprietary to companies. These restrictions hinder the ability to conduct studies using real-world data for defect classification on wafers, thereby limiting the generalizability and applicability of the research findings. Therefore, the research problem at hand revolves around addressing the challenges of limited public datasets, class imbalance issues, and restricted access to real-world data to develop effective deep learning models for early defect classification in semiconductor wafer production. Overcoming these challenges will contribute to improving quality control processes, reducing manufacturing costs, and enhancing overall productivity in the semiconductor industry. The challenges of deep-learning-based classification models for wafer map defects including the limited availability of publicly accessible datasets, the importance of proper data preparation, the need to classify mixed-type defects accurately, the pursuit of high classification performance, and the necessity to address performance issues such as memory constraints during data preprocessing, training, and model evaluation.

The key research contributions of our study are:

1. Proposed CNN Model Based on GoogLeNet: Our research introduces a novel CNN model based on the GoogLeNet architecture for wafer defect pattern classification. The proposed GoogLeNet model provides a strong foundation for accurate and robust classification of wafer defect patterns.
2. Classification of Single and Mixed Wafer Defect Patterns: Our proposed CNN model is designed to classify both single and mixed types of wafer defect patterns. This allows for comprehensive and reliable classification, even when multiple defect types are present on the same wafer.
3. Generalization to New Defect Types: Our proposed CNN model demonstrates promising generalization capabilities to new or unseen defect types. While training on a specific set of defect patterns, the model's underlying architecture and learned features enable it to potentially classify previously unseen defect types with reasonable accuracy.
4. Achievement of High Accuracy: Through extensive experimentation and evaluation, our proposed CNN model achieves an impressive average accuracy of 99.9% in wafer defect pattern classification. This high accuracy demonstrates the effectiveness and reliability of our approach in accurately identifying and classifying different types of wafer defects.
5. Publication of WM-300K+ Wafer Map Dataset: In addition to proposing a novel CNN model, we also created a new dataset called "WM-300K+ wafer map [Single & Mixed]." This dataset is noteworthy as it is cleaned, and balanced, and consists of more than 300,000 wafer map images with 36 classes. We publish this dataset on Kaggle, providing a valuable resource for the research community and enabling further advancements in wafer map analysis and classification.
6. Utilization of CUDA for Enhanced Training and Testing Speed: To speed up the training and testing process of our proposed CNN model, we utilize CUDA, a parallel computing platform that enables significant performance improvements when training deep neural networks.

2. LITERATURE REVIEWS

Several previous studies have focused on wafer defect classification, each employing different datasets, CNN architectures, and training techniques. A comprehensive overview of these studies is provided in Table 1. The table includes details such as the dataset used, the number of defect patterns analyzed, whether single or mixed defects were classified, the specific CNN architecture utilized in each study, and the corresponding test accuracy achieved for classification.

Table 1: A Comprehensive Overview of The Performance of Different CNN Models in Defect Classification.

Ref	Dataset used	Single or Mixed defects classification	Accuracy
[19]	WM-811K	Single	93.25%
[20]	WM-811K and MixedWM38	Single and Mixed	93.6% for single and 97.5% for mixed
[21]	WM-811K	Single	96.93%
[22]	Dataset-TT	Single	96.2%
[23]	MixedWM38	Single and Mixed	95.8%
Our work	Preprocessed WM-811K	Single and Mixed	99.9%

Researchers have developed custom CNN architectures for classifying wafer defects, aiming to improve performance and efficiency. In [19], a 13-layer CNN was proposed to detect and classify eight known wafer map defects. The authors applied noise removal and resizing techniques to the wafer map images. A Dropout method with a probability of 0.5 was used for training regularization. The detection accuracy achieved an impressive rate of 99.98%, while classification yielded an average accuracy of 93.25%. Among specific defect types, 'Donut' had a minimum average accuracy rate of 86%, whereas both 'Edge-Ring' and 'Near-Full' defects achieved maximum accuracies at around 100%.

Researchers have studied the use of existing CNN architectures like 'PeleeNet' and 'ShuffleNet-v2' for wafer defect classification. A lightweight classifier called "WM-PeleeNet" was proposed by [20], using a derived CNN architecture with nine layers to achieve a balance between accuracy and efficiency. The researchers utilized two datasets: 'WM-811K' and 'MixedWM38'. Data augmentation techniques were employed to address dataset imbalance, including convolutional autoencoder, GAN-based, and image transformation methods. Experimental results showed an average accuracy of 93.6% for single-wafer defect patterns and 97.5% for mixed types of wafer defects using the 'MixedWM38' dataset.

In [21], a silicon wafer defect identification and classification model were proposed by researchers. This model consists of a pre-trained deep transfer learning model called ShuffleNet-v2 with seven layers using CNN architecture. The model achieves an overall accuracy of 96.93%, precision of 95.40%, recall of 96.26%, and F1-score of 95.75\% in classifying the defects. The training and testing phase utilized the 'WM-811K' wafer dataset, with data augmentation performed using a six-layer convolutional autoencoder CNN model. The total number of images used after data augmentation was 19,707, representing nine different patterns for wafer defects. However, because it is focused on being lightweight, the ShuffleNet-v2 may sacrifice some accuracy to achieve faster inference time and lower computational cost.

Researchers proposed an automatic defect classification system for wafer defect identification and classification. In [22] the researchers present a system that utilizes DL techniques, specifically a ResNet101-based CNN model that was pre-trained on the 'ImageNet' dataset. For the sizing classification of two specific classes, the system employs a single-shot detection architecture based on VGG-16. The research utilized a dataset consisting of 8 defect classes, with 2 subclasses representing similar defects but varying in sizes. Data augmentation techniques were implemented to expand the dataset size. The obtained results showed a top-1 accuracy of 91.1% and a top-3 accuracy rate of 96.2%, with each class being correctly classified at least 69% of the time.

Other researchers have employed semantic segmentation in image processing and computer vision by dividing an image into regions and assigning a semantic label to each region using CNN. In [23], the researchers proposed a new framework for segmenting defect patterns on wafer maps when multiple defect types are present on the same wafer. They used a 'U-NET' CNN architecture and achieved high accuracy on both synthetic and 'MixedWM38' datasets, addressing the challenge of identifying complex defect patterns arising from mixed-type defects.

Neither of the previous studies specifically addressed the use of parallel programming techniques like CUDA to improve performance in wafer defect classification. Many researchers have resized images to 224x224 or 416x416 when dealing with the 'WM-811K' dataset, despite the original dataset having maximum counts at wafer image dimensions of 27x25. This resizing process can lead to loss of fine-grained details and increased computational resources such as memory and CPU consumption, which may impact model performance and training/testing times. Some researchers use computationally expensive methods for data augmentation, such as convolutional autoencoder in a GAN-based architecture.

3. DATASET

In this study, a dataset called "WM-811k" was used to derive a new balanced dataset. This dataset is available on the Kaggle website which consists of nine distinct patterns of wafer defects [24]. Notably, the "waferMap" column in the dataset represents wafer map images. These images are stored as two-dimensional arrays, with each pixel being represented by an 8-bit unsigned integer, as shown in Figure 3.

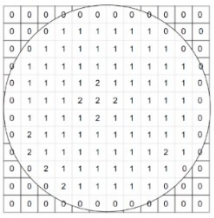


Figure 3: Visualization of wafer map representation with color map value from a dataset.

The color map used for visualizing the wafer map image comprises three values according to Equation 1 that help in identifying the defect patterns and their distribution in the wafer map, which is crucial for accurate defect classification using CNN models.

$$\begin{cases} 0 \rightarrow Non-Die\ Area \\ 1 \rightarrow Die\ Pass\ Test\ Successfully \\ 2 \rightarrow Die\ Contain\ Defect \end{cases} \quad (1)$$

The dataset contains a total of 811,457 wafer maps. However, it is worth noting that approximately 79% of these wafer maps do not have any pattern label for defect type or have "none" pattern label. A total of 25,519 wafer maps in the dataset have defect pattern labels. Additionally, by examining the frequency distribution of defect patterns in the dataset for wafer maps with defect pattern labels, we can observe significant variations across different types of defect patterns. Moreover, it is important to highlight that the wafer maps arrays exhibit dimensions spanning across 632 unique values. Consequently, considering these variations in both defect pattern frequencies and wafer map dimensions becomes crucial during the data pre-processing phase to ensure accurate classification of defects using our proposed model.

4. METHODOLOGY

4.1 Data Preprocessing and Augmentation

Data preprocessing is crucial for preparing the dataset for analysis and modelling to ensure accurate and reliable results, especially when dealing with variations in defect pattern frequencies and wafer map dimensions. Preprocessing methods involve data cleaning, extraction, converting wafer map images to RGB format, and resizing these images to 56 x 56 dimensions. The phase includes multiple steps as shown in Figure 4, while additional steps are for preparing data for training and testing.

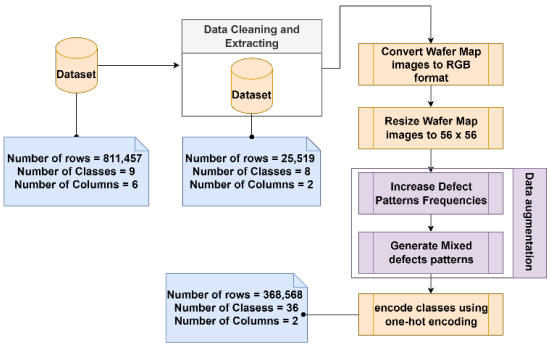


Figure 4: Illustration of the main steps in data preprocessing, data augmentation, and encoding class labels, outlining the essential pre-processing steps required for training a CNN model for defect classification.

During dataset preprocessing for our deep learning model, we addressed classes with no defects or inaccurate defect patterns (failure type). These classes were excluded from further analysis to prioritize authentic defect patterns and ensure the quality of the dataset. This refinement facilitated more effective training for our deep learning model, resulting in 25,519 wafer map images representing genuine defect patterns being retained after removing 785,938 wafer map images. In addition, the wafer map images in the dataset, which are represented in grayscale format, were mapped to three specific values: 0, 127, or 255. This mapping was done according to Equation 2, where these values corresponded to the minimum, median, and maximum pixel intensities, respectively.

$$pixel\ value = \begin{cases} 0 \rightarrow 0 \\ 1 \rightarrow 127 \\ 2 \rightarrow 255 \end{cases} \quad (2)$$

The original wafer map images had low pixel values, resulting in predominantly black images. To improve visualization, the values were mapped to a wider range (0, 127, 255) representing black, mid-grey, and white respectively. The grayscale images were then converted to RGB format for better analysis and defect identification. The dataset had varying dimensions, so a Bicubic interpolation was used to resize the images to 56x56. This involved averaging the neighboring pixels [25], [26].

Data augmentation techniques involve applying transformations and modifications to existing data to create new samples to improve generalization and help CNN models handle unseen data while reducing overfitting [27]. Two approaches of data augmentation techniques were utilized to enhance the performance and accuracy of the classification model. The first approach involved increasing the frequency of defect patterns by applying various transformations such as random rotation, random horizontal flip, and random vertical flip. This was done to address the class imbalance in existing defect patterns and improve the model's ability to generalize and perform well on unseen data [28], [29]. The second approach involved creating mixed types of wafer defects by combining different defect patterns. This comprehensive collection of training data enhances the representation of various possible patterns that a model may encounter during practical applications, ultimately improving the model's ability to generalize and perform well on new data [28], [30]. The maximum pixel value from all input images at the same position is used since the pixel value representing a defect has a maximum value of 255. Pseudocode 1 presents the pseudocode for a mixed defect patterns generator algorithm. This algorithm takes a list of input images as input and generates a mixed image by setting each pixel value of the mixed image to the maximum pixel values of all input images at each corresponding location. The resulting mixed images will have the same dimensions as the input images and will contain the highest pixel values from each input image.

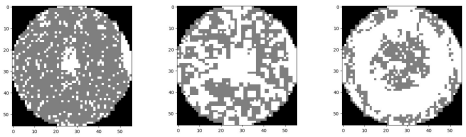
Pseudocode 1 mixed defect patterns generator algorithm.

Input: input images ($input_{images}$)
1: Create a new blank image M to hold the mixed image pixels with the same dimensions as the first image of $input_{images}$
2:For x in the range of [0, $M\ width - 1$]:
3: For y in the range of [0, $M\ height - 1$]:
4: Obtain the maximum pixel value $pixel_{\max value}$at (x, y) for all $input_{images}$.
5: Set $M[x, y]$ to $pixel_{\max value}$
6: End For
7:End For
8:Return M

Overall, these two approaches of data augmentation techniques proved to be effective in improving the performance and accuracy of the classification model. By increasing the size and diversity of the training dataset, the model was better equipped to handle unseen data and achieve higher accuracy rates [31]. Also, these approaches allow for the exploration and generation of novel defect patterns that may occur in real-life situations. For instance, by combining two types of defects, new defect patterns can be created such as Center with Edge-Loc(C+EL), Center with Edge-Ring(C+ER), Center with Loc(C+L), Center with Scratch(C+S), Loc with Scratch(L+S), Donut with Scratch(D+S), Donut with Edge-Loc(D+EL), Donut with Edge-Ring(D+ER), Donut with Loc(D+L), Edge-Loc with Loc(EL+L), Edge-Loc with Scratch(EL+S), Edge-Ring with Loc(ER+L) and Edge-Ring with Scratch (ER+S).

Similarly, the combination of three defects yields additional new defect patterns such as: Center with Edge-Loc with Scratch(C+EL+S), Center with Edge-Ring with Scratch(C+ER+S), Center with Edge-Loc with Loc(C+EL+L), Center with Edge-Ring with Loc(C+ER+L), Center with Loc with Scratch(C+L+S), Donut with Edge-Loc with Scratch(D+EL+S), Donut with Edge-Ring with Scratch(D+ER+S), Donut with Edge-Loc with Loc (D+EL+L), Donut with Edge-Ring with Loc(D+ER+L), Donut with Loc with Scratch(D+L+S) and Edge-Loc with Loc with Scratch(EL+L+S). Furthermore, when four defects are combined, it leads to the creation of even more new defect patterns such as: Center with Loc with Edge-Loc with Scratch(C+L+EL+S), Center with Loc with Edge-Ring with Scratch(C+L+ER+S), Do-nut with Loc with Edge-Loc with Scratch(D+L+EL+S) and Donut with Loc with Edge-Ring with Scratch(D+L+ER+S).

In total, a collection of 28 new mixed defect patterns can be generated, in addition to the 8 single defect patterns. These mixed defect patterns provide a broader representation of the possible defect variations that may be encountered in practical applications. Figure 5 shows three samples of mixed defect patterns. By incorporating these mixed defect patterns into the dataset, the model can be trained to recognize and classify a wider range of defect types and combinations. After applying data augmentation, the number of wafer map images increased to a total of 368,568 images. Each defect pattern constitutes approximately 3% of the dataset's entirety out of these images, encompassing a combined total of 36 unique defect patterns for both single and mixed types, the new dataset is called “WM-300K+ wafer map [Single & Mixed]”. Furthermore, an approach utilizing one-hot encoding was employed to encode the defect patterns where each defect pattern is represented by a binary vector with values of 1 or 0 based on its presence [32].



(a) (b) (c)

Figure 5: Examples of Mixed Defect Patterns include (a) Center with Edge-Loc, (b) Center with Edge-Loc with Scratch, and (c) Donut with Loc with Edge-Loc with Scratch.

4.2 The Proposed Classification Framework

An automatic wafer defects classification framework using a CNN deep learning model is presented to address the gaps in the literature discussed in Section 3 and to fulfil the study's aims and objectives. The Classification Framework Implementation utilizes PyTorch, an efficient open-source Python framework for machine learning. PyTorch supports GPU-accelerated tensor computing and automatic differentiation, making it suitable for deep neural networks. It employs data loaders to handle dataset loading and batching, enabling efficient training on large datasets [33]. After performing data pre-processing and data augmentation techniques to tackle the data imbalance issues and create new mixed defect patterns for further study, the methodology involves generating a new dataset and making it publicly available for use by researchers.

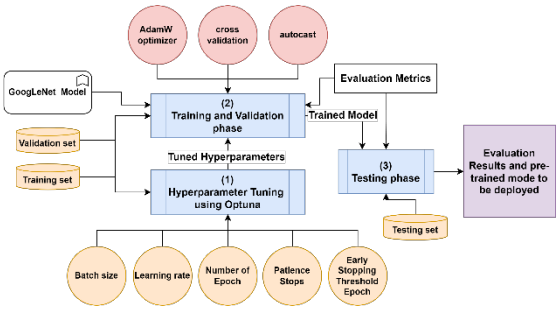


Figure 6: The Proposed Classification Framework.

Thereafter Optuna is used to tune optimal hyperparameters. The hyperparameters and the new dataset are used for training and validating the modified GoogLeNet model. "Auto cast" mixed precision feature is utilized during the training and validation to automatically cast certain operations to lower-precision data types, like half-precision, while keeping other operations in higher-precision data types, such as single-precision. This allows for faster computation and reduced memory requirements without sacrificing model accuracy [34]. Also, Adam with Weight Decay Regularization is used for optimizing the model's performance by adjusting the weights and biases based on computed gradients.

K-fold cross-validation and early stopping are used to improve generalization and prevent overfitting during training. With k-fold cross-validation, the dataset is split into k equally sized folds. The model is then trained and evaluated k times, with each fold serving as the validation set once while the remaining folds are used for training [35]. Early stopping involves monitoring the model's performance on a validation set during training and stopping early if the performance starts to degrade. Various evaluation metrics, such as accuracy, precision, recall and F1 score, are employed to comprehensively assess the performance of the model in both the training and testing phases.

4.3 The Modified GoogLeNet Model

In 2015, Google developed GoogLeNet, a deep convolutional neural network for image classification. It utilizes multiple convolutional layers with varying filter sizes and pooling operations to extract features at different scales. GoogLeNet includes auxiliary classifiers as intermediate layers to address the vanishing gradient problem and improve accuracy by reducing overfitting. The computational efficiency is achieved through inception modules for feature extraction [36]. Although the original GoogLeNet architecture was designed for images with dimensions of 224 x 224 and 1024 classes, it has been modified to be suitable for images with dimensions of 56 x 56 pixels and 36 classes by adding a new fully connected layer at the end of the GoogLeNet layers that maps the 1024 classes to 36 classes.

The modified GoogLeNet architecture consists of 19 layers, including convolutional and max-pooling layers as visualized in Figure 7. It also incorporates inception modules, an average pooling layer, a dropout layer, and a linear layer for the final output. The "BasicConv2d" layer is responsible for extracting features through convolutional operations from the input data. Additionally, there's the "MaxPool2d" to reduce spatial dimensions and multi-branch convolutional blocks known as Inception modules that aid in capturing different scales and types of features. Lastly, there's an "AvgPool2d" layer to apply average pooling to the input feature maps before using a dropout preceding the fully connected last layer designed to map input features to output classes while preventing overfitting [16], [37], [38].

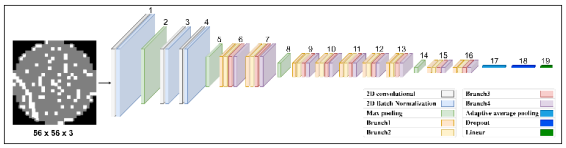


Figure 7: The visualization of the modified GoogLeNet model architecture.

The architecture is designed to take an input of shape [512, 3, 56, 56] and produce an output of shape [512, 36] as illustrated in Table 2. Notably, the number 512 here presents the batch size, the number 3 presents the number of input image channels and the numbers 56,56 represent the input image dimensions. All the convolutions, including the convolutions inside the inception module, use rectified linear activation [38].

Table 2: Description Of Layers in The Modified GoogLeNet Model with Input and Output Shapes.

#	Layer Name	Input Shape	Output Shape
1	BasicConv2d	[512, 3, 56, 56]	[512, 64, 28, 28]
2	MaxPool2d	[512, 64, 28, 28]	[512, 64, 14, 14]
3	BasicConv2d	[512, 64, 14, 14]	[512, 64, 14, 14]
4	BasicConv2d	[512, 64, 14, 14]	[512, 192, 14, 14]
5	MaxPool2d	[512, 192, 14, 14]	[512, 192, 7, 7]
6	Inception 1	[512, 192, 7, 7]	[512, 256, 7, 7]
7	Inception 2	[512, 256, 7, 7]	[512, 480, 7, 7]
8	MaxPool2d	[512, 480, 7, 7]	[512, 480, 3, 3]
9	Inception 3	[512, 480, 3, 3]	[512, 512, 3, 3]
10	Inception 4	[512, 512, 3, 3]	[512, 512, 3, 3]
11	Inception 5	[512, 512, 3, 3]	[512, 512, 3, 3]
12	Inception 6	[512, 512, 3, 3]	[512, 528, 3, 3]
13	Inception 7	[512, 528, 3, 3]	[512, 832, 3, 3]
14	MaxPool2d	[512, 832, 3, 3]	[512, 832, 2, 2]
15	Inception 8	[512, 832, 2, 2]	[512, 832, 2, 2]
16	Inception 9	[512, 832, 2, 2]	[512, 1024, 2, 2]
17	AvgPool2d	[512, 1024, 2, 2]	[512, 1024, 1, 1]
18	Dropout	[512, 1024]	[512, 1024]
19	Fully connected	[512, 1024]	[512, 36]

The weights of the modified GoogLeNet model will be initialized via loading the pre-trained weights of the original GoogLeNet model that have already captured relevant information from the training data from a dataset called "ImageNet". The utilization of a pre-trained GoogLeNet model and weight initialization from it provides a strong foundation for our deep learning architecture. By incorporating this approach, we aim to enhance the performance and efficiency of our model while reducing the need for extensive training on the dataset [39].

4.4 Hyperparameter Tuning and Optimization Technique

Hyperparameters are parameters that govern the learning process and dictate the values of model parameters acquired by a learning algorithm [40]. Hyperparameters are settings or configurations that are not learned from the data but are set before training the model such as batch size, learning rate, number of epochs and patience stop as illustrated in Table 3. Optuna which is an open-source hyperparameter optimization framework is selected to find the optimal hyperparameters which have a significant impact on the model's performance [41].

Table 3: Description Of Layers in The Modified GoogLeNet Model with Input and Output Shapes.

#	Hyper Parameter Name	Hyper Parameter Description
1	Batch size	The batch size refers to the number of training examples that are used in both the forward and backward passes of a neural network during its training phase.
2	Learning rate	Determines the step size at which the optimizer adjusts the weights of the model during training.
3	Number of Epochs	The epoch refers to one complete pass through the entire training dataset during the training process.
4	Patience Stops	Number of epochs without improvements to wait before early stopping
5	Early Stopping Threshold Epoch	The minimum number of epochs that must be completed before early stopping is checked.

The Optuna is based on the Bayesian Optimization Algorithm which is used to find the global maximum or minimum solution of a scaler objective function in a bounded domain $(f: \mathbb{R}^d \rightarrow \mathbb{R})$. It aims to model the objective function f to specify its distribution. For a set of points $x \in \mathbb{R}^d$, the evaluation of membership of the objective function f is calculated by the following equation [42]:

$$x_{new} = max_{x \in \mathbb{R}^d} f(x) \quad (3)$$

Optimizers are crucial for adjusting model weights and learning rates to minimize error or maximize production efficiency [43]. Gradient Descent is an iterative technique that modifies parameters to reduce a given convex function, achieved by moving in the opposite direction of the steepest ascent determined by the learning rate

using derivatives [43]. Adam with Weight Decay Regularization is an example optimizer introduced in 2019 as a modification of the Adaptive Moment Estimation algorithm, aiming to improve weight decay behaviour and incorporate adaptive learning rates [44]. The goal of employing AdamW optimizer is to minimize the cost function value as much as possible. AdamW integrates weight decay directly into its framework without affecting adaptive learning rates, resulting in enhanced performance for generalization and improved convergence properties [45].

4.5 Evaluation Metrics

Evolutionary metrics were used to evaluate the proposed approach, including confusion matrix, accuracy, F1 score, precision, recall and Receiver Operating Characteristics. The inclusion of a confusion matrix helps assess the classification performance of the CNN model by comparing true and predicted values. In this matrix, rows correspond to true values while columns represent predicted values. The assessment yields four possibilities: true positive, false positive, true negative and false negative [46]. Accuracy refers to the extent to which the model effectively categorizes all instances within a dataset. It is computed by dividing the total number of correct predictions by the overall number of predictions made [47]. The accuracy is calculated by the following equation:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (4)$$

In this context, TP represents true positives, FP represents false positives, TN represents true negatives, and FN represents false negatives.

The F1 score is determined by computing the harmonic mean of precision and recall. Precision measures the proportion of correctly predicted positive instances out of all predicted positive instances, while recall gauges how many actual positive instances are accurately identified as positive [48]. The F1 score is calculated by the following equation:

$$F1\ score = 2 * \frac{(precision * recall)}{(precision + recall)} \quad (5)$$

The Receiver Operating Characteristic (ROC) curve is a metric used to evaluate the performance of a CNN model in class discrimination. It measures the ability of the model to accurately classify different classes by analyzing true positive rate (TPR) and false positive rate (FPR) across various threshold values for classification. It plots the TPR against the FPR at various threshold settings [49].

The formulas for TPR and FPR are as follows:

$$TPR = \frac{TP}{(TP + FN)} \quad (6)$$

$$FPR = \frac{FP}{(FP + FN)} \quad (7)$$

The shape and position of the ROC curve provide insights into the model's performance, with proximity to the top-left corner indicating better performance due to higher TPR and lower FPR. A curve close to the diagonal line suggests random guessing, while a curve below it indicates poor performance in distinguishing between positive and negative cases.

5. RESULTS AND DISCUSSIONS

The proposed model was trained and validated using the training and validation dataset where the dataset was split into 80% for training and 20% for testing. Subsequently, the training set was further divided into 70% for training and 30% for validation. This division allowed for training the model on a subset of the data while validating its performance on another subset. The hyperparameters values were used after tunning as illustrated in Table 4.

Table 4: Tunned Hyperparameter Optimal Values

#	Hyper Parameter Name	Hyper Parameter Value
1	Batch size	512
2	Learning rate	0.0008
3	Number of Epochs	100
4	Patience Stops	5
5	Early Stopping Threshold Epoch	15

5.1 Experimental Training Results

This study applies a stratified 6-fold cross-validation in which they have the same proportion of class distribution. As shown in Table 5, the average training accuracy across all folds is 99.40%, indicating that the model performs well on the training data. The average validation accuracy is 97.80%, suggesting that the model generalises reasonably well to unseen data. In addition, the average training loss is 0.013, which indicates that the model's predictions are close to the actual values during training and the average validation loss is 0.044, indicating that the model's predictions are slightly less accurate on the validation data compared to the training data. Overall, these results suggest that the proposed CNN model is effective in classifying wafer defects.

Journal of Theoretical and Applied Information Technology
15th February 2024. Vol.102. No 3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 5: Cross-Validation Results, Detailing the Average Training and Validation Accuracy, As Well As Training and Validation Loss for All Folds in A CNN-Based Defect Classification Model

Average Training Accuracy (%)	Average Validation Accuracy (%)	Average Training Loss	Average Validation Loss
99.40	97.80	0.013	0.044

The new dataset was used to train and validate the ShuffleNetV2 and ResNet-50 CNN models. The Modified GoogLeNet achieved the highest training accuracy of 99.40% and a validation accuracy of 97.80%, with the lowest training loss of 0.013 and a moderate validation loss of 0.044. ShuffleNetV2 and ResNet-50 showed slightly lower performance compared to Modified GoogLeNet. Overall, Modified GoogLeNet demonstrated the highest training accuracy and good generalization performance on the validation set.

5.2 Model Evaluation

The confusion matrix as shown in Figure 8 highlights strong performance in classes like "C+EL," "C+EL+L," "C+EL+S," and "C+ER," with high true positives and true negatives. It also shows that the class "Near-full" demonstrates effective detection with very few false positives and false negatives, indicating high accuracy. However, classes like "Edge-Loc" and "Loc" show a higher number of false negatives compared to other classes, suggesting room for improvement in accurately classifying these instances. Further enhancements may be needed to improve accuracy in classifying instances for these specific classes.

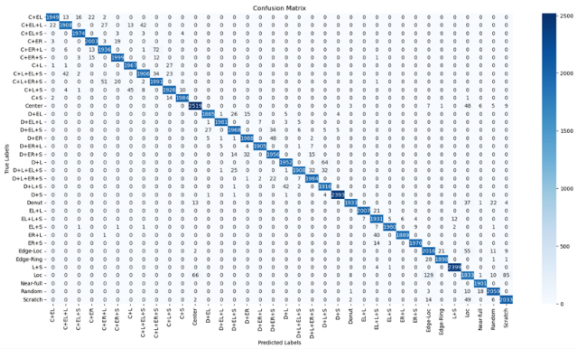


Figure 8: Confusion Matrix for all classes in a CNN-based defect classification model.

Table 6 shows that the classification algorithm achieves high average accuracy, precision, recall, and F1 scores of 99.99%, 0.97, 0.97, and 0.97 respectively for all 36 wafer defect patterns. It correctly classifies 99.9% of instances, demonstrating a high level of overall correctness. With a precision and recall of 0.97, it shows a low rate of false positives and false negatives, indicating effective and accurate identification and classification across all classes. The classification algorithm demonstrates a good balance between precision and recall, indicating overall robust performance with an F1 score of 0.97. These high values for accuracy, precision, recall, and F1 score suggest that the classification algorithm is effective, and accurate in identifying and classifying instances across all classes.

Table 5: Average Of Performance Metrics for All Classes in A CNN-Based Defect Classification Model.

Average Accuracy (%)	Average Precision	Average Recall	Average F1 Score	Average ROC AUC
99.99%	0.97	0.97	0.97	0.99

The TPR and FPR for 36 classes were used to compute the ROC curve for each class. As shown in Figure 9, the ROC curve for all classes is closer to the top-left corner of the plot which indicates that the model can accurately classify positive cases while minimizing false positives. The majority of classes show high true positive rates, many surpassing 0.98, and a low false positive rate (3.9045e-04). However, the "Loc" class has a lower true positive rate of approximately 0.8598, suggesting potential challenges for accurate differentiation that may require further fine-tuning or optimization efforts.

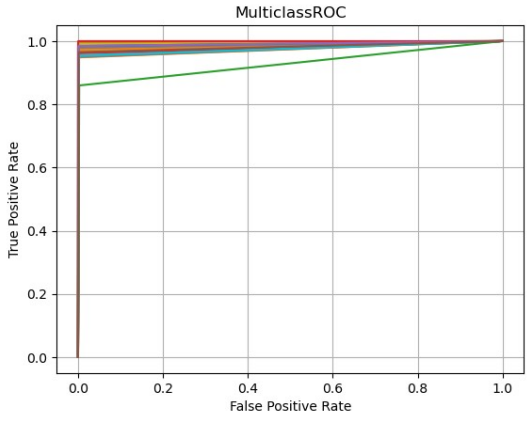


Figure 9: ROC Curves for all Defect classes.

The other trained models are evaluated using the same testing set. The ResNet-50 reached 99.9% accuracy, and ShuffleNetV2 achieved 99.8% accuracy. In terms of precision, recall, and F1 score, all models consistently performed well, with an average of 0.97 for ResNet-50 and an average of 0.96 for ShuffleNetV2. This indicates successful

classification of positive cases while minimizing false positives (precision), capturing true positive cases (recall), and achieving a balanced trade-off between precision and recall (F1 score).

Memory limit errors were encountered during the data preprocessing and model training stages due to inadequate memory for handling the data or computations. Our study focused on optimizing memory usage during data preprocessing and model training. We encountered memory limit errors due to inadequate memory resources, which hindered our progress. Various strategies were employed to overcome these errors. [51], [52].

By using generators and data loaders, we were able to efficiently handle large datasets within limited memory resources. Freeing up unnecessary variables and employing memory-efficient data structures further contributed to efficient memory utilization. Batch processing allowed for efficient training, while the auto-cast feature reduced computational resource requirements and enabled the use of larger batch sizes. Leveraging the CUDA parallel computing platform provided an opportunity to harness the power of NVIDIA GPUs for general-purpose computing tasks, further optimizing resource allocation. The PyTorch allows to use of CUDA devices with simple APIs to transfer data to GPU memory and perform operations on GPU. Also, the training and validation processes are performed on GPU [53]. The Modified GoogLeNet model achieved an average CPU usage of 35.17% and a maximum memory usage of 19469.69 MB during execution on the CPU. When executed on the GPU, the model achieved an average GPU usage of 81.30% and a maximum CUDA memory usage of 6812.47 MB. The execution time on the GPU was significantly faster, taking only 4,641.36 seconds. The speedup achieved by using the GPU instead of the CPU is approximately 38.78x, indicating a significant performance improvement.

6. CURRENT PROBLEMS AND OPEN RESEARCH ISSUES

In the context of wafer map defects classification using CNN, there are several current problems, challenges, and open research issues that can be addressed. Some of these include:

1. Limited Availability of Public Datasets: Access to high-quality, diverse, and publicly available datasets for training and evaluating CNN models for wafer map defect classification is limited. Developing standardized benchmark datasets would be beneficial for researchers and practitioners in this field.
2. Mixed-Type Defect Classification: Classifying mixed-type defects (e.g., scratches, particles, and pattern defects) accurately remains a challenge. Research is needed to develop CNN models that can effectively handle the classification of multiple defect types within a single wafer map.
3. Performance Improvement: There is a continuous need to enhance the classification performance of CNN models for wafer map defect detection. This includes improving accuracy, reducing false positives, and addressing issues related to model generalization.
4. Data Preprocessing and Memory Constraints: Addressing memory constraints during data preprocessing, model training, and evaluation is crucial, especially when dealing with large-scale wafer map datasets. Efficient data preprocessing techniques and memory-optimized model architectures are areas for further exploration.
5. Interpretability and Explainability: Developing methods to interpret and explain the decisions made by CNN models in the context of wafer map defect classification is an open research issue. Understanding how the model arrives at its classifications is essential for gaining trust and acceptance in real-world applications.

Addressing these problems and research issues will contribute to the advancement of CNN-based wafer map defect classification, leading to more robust and reliable defect detection systems in semiconductor manufacturing.

7. FUTURE RESEARCH DIRECTIONS

While our research makes significant contributions to the field of wafer defect pattern classification, certain limitations should be acknowledged:

1. Dependency on Dataset Quality: The effectiveness of our approach is highly dependent on the quality and diversity of the training dataset. If the dataset contains inaccuracies, noise, or biases, it may impact the model's performance and generalizability. Therefore, ensuring a high-quality and representative dataset is crucial for obtaining reliable results.
2. Computational Resource Requirements: Our proposed CNN model, particularly when using

the GoogLeNet architecture and CUDA for enhanced speed, may require significant computational resources during training and testing. This includes high-performance GPUs and sufficient memory capacity. Researchers with limited access to such resources may face challenges in replicating our experiments or applying our approach in resource-constrained environments.

3. Data Augmentation Limitations: While data augmentation can help address dataset issues such as class imbalance, the effectiveness of this technique may vary depending on the specific characteristics of the dataset. In some cases, data augmentation may not fully mitigate the challenges associated with imbalanced data, leading to potential biases or limitations in the model's performance.
4. Lack of Evaluation on Real Wafer Map Images: Our proposed model has not been evaluated on real wafer map images. Although we achieved high accuracy using our dataset, the model's performance on real-world wafer map images may differ because of variations in image quality, noise, and other factors specific to real production environments. Further evaluation and validation of real wafer map images are necessary to assess the model's practical applicability.

Several areas warrant further investigation and exploration. Future work should focus on addressing the following aspects:

1. Expand the dataset to include a wider range of defect types and variations, enabling the model to handle a broader array of real-world scenarios.
2. Optimize computational resource requirements by developing more efficient architectures or exploring alternative hardware configurations that can achieve comparable performance with reduced resource demands.
3. Further evaluation and validation of real wafer map images are necessary by conducting extensive evaluations to understand the model's performance under these realistic conditions and identify any necessary adaptations or improvements.

8. CONCLUSIONS

This paper presents an enhanced CNN model based on the GoogLeNet architecture for wafer defect pattern classification, achieving an impressive average accuracy of 99.9%. Additionally, the paper contributes to the field by introducing the "WM-300K+ wafer map [Single & Mixed]" dataset and leveraging CUDA for enhanced training and testing speed, surpassing existing benchmarks and achieving state-of-the-art results in wafer defect pattern classification. However, limitations were identified, including dependence on dataset quality, computational resource requirements, and data augmentation constraints. Notably, the proposed model has not been evaluated on real wafer map images, indicating the need for further assessment to evaluate its practical applicability. Future work should prioritize addressing these limitations by enhancing dataset quality, optimizing computational resource requirements, improving data augmentation techniques, and evaluating the model on real wafer map images. Overall, the study presents a powerful CNN model capable of accurately classifying both single and mixed wafer defect patterns, surpassing previous studies in accuracy, with significant potential to enhance the efficiency and effectiveness of wafer defect analysis in semiconductor manufacturing.

REFERENCES:

[1] Y. Q. Chen, B. Zhou, M. Zhang, and C. M. Chen, "Using IoT technology for computer-integrated manufacturing systems in the semiconductor industry," Applied Soft Computing Journal, vol. 89, 2020, doi: 10.1016/j.asoc.2020.106065.

[2] A. A. R. M. A. Ebayyeh and A. Mousavi, "A Review and Analysis of Automatic Optical Inspection and Quality Monitoring Methods in Electronics Industry," IEEE Access, vol. 8. 2020. doi: 10.1109/ACCESS.2020.3029127.

[3] Fortune Business Insights, "Consumer Electronics Market Size to Hit USD 989.37 Billion." Accessed: Aug. 12, 2023. [Online]. Available: https://www.globenewswire.com/news-release/2023/02/28/2616731/0/en/Consumer-Electronics-Market-Size-to-Hit-USD-989-37-Billion-by-2027-At-5-3-CAGR.html

[4] K. T. Turner and S. M. Spearing, "Mechanics of direct wafer bonding," Proceedings of the Royal Society A: Mathe-matical, Physical and Engineering Sciences, vol. 462, no. 2065, 2006, doi: 10.1098/rspa.2005.1571.

[5] J. N. D. Gupta, R. Ruiz, J. W. Fowler, and S. J. Mason, "Operational planning and control of semiconductor wafer production," Production

Planning and Control, vol. 17, no. 7, 2006, doi: 10.1080/09537280600900733.

[6] K. Kyeong and H. Kim, "Classification of Mixed-Type Defect Patterns in Wafer Bin Maps Using Convolutional Neural Networks," IEEE Transactions on Semiconductor Manufacturing, vol. 31, no. 3, 2018, doi: 10.1109/TSM.2018.2841416.

[7] Semiconductor Industry Association, "Chipmakers Are Ramping Up Production to Address Semiconductor Shortage. Here's Why that Takes Time," Feb. 2021, Accessed: Aug. 12, 2023. [Online]. Available: https://www.semiconductors.org/chipmakers-are-ramping-up-production-to-address-semiconductor-shortage-heres-why-that-takes-time/

[8] R. Desineni and E. Tuv, "High-Value AI in Intel's Semiconductor Manufacturing Environment," Intel. Intel, 2018. Accessed: Oct. 26, 2023. [Online]. Available: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/ai-in-semiconductor-manufacturing-paper.pdf

[9] T. Yuan, S. Z. Ramadan, and S. J. Bae, "Yield prediction for integrated circuits manufacturing through hierar-chical bayesian modeling of spatial defects," IEEE Trans Reliab, vol. 60, no. 4, 2011, doi: 10.1109/TR.2011.2161698.

[10] U. Kaempf, "The Binomial Test: A Simple Tool to Identify Process Problems," IEEE Transactions on Semiconductor Manufacturing, vol. 8, no. 2, 1995, doi: 10.1109/66.382280.

[11] J. Yu and X. Lu, "Wafer Map Defect Detection and Recognition Using Joint Local and Nonlocal Linear Discrimi-nant Analysis," IEEE Transactions on Semiconductor Manufacturing, vol. 29, no. 1, 2016, doi: 10.1109/TSM.2015.2497264.

[12] M. Piao, C. H. Jin, J. Y. Lee, and J. Y. Byun, "Decision tree ensemble-based wafer map failure pattern recognition based on radon transform-based features," IEEE Transactions on Semiconductor Manufacturing, vol. 31, no. 2, 2018, doi: 10.1109/TSM.2018.2806931.

[13] S. Parrish, "A Study of Defects in High Reliability Die Sort Applications," International Symposium on Microelec-tronics, vol. 2019, no. 1, 2019, doi: 10.4071/2380-4505-2019.1.000463.

[14] M. Liukkonen and Y. Hiltunen, "Recognition of Systematic Spatial Patterns in Silicon Wafers Based on SOM and K-means," 2018. doi: 10.1016/j.ifacol.2018.03.075.

[15] T. Tiwari, T. Tiwari, and S. Tiwari, "How Artificial Intelligence, Machine Learning and Deep Learning are Radi-cally Different?," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 8, no. 2, 2018, doi: 10.23956/ijarcsse.v8i2.569.

[16] J. Schmidhuber, "Deep Learning in neural networks: An overview," Neural Networks, vol. 61. 2015. doi: 10.1016/j.neunet.2014.09.003.

[17] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and appli-cation in radiology," Insights into Imaging, vol. 9, no. 4. 2018. doi: 10.1007/s13244-018-0639-9.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, 1986, doi: 10.1038/323533a0.

[19] N. Yu, Q. Xu, and H. Wang, "Wafer defect pattern recognition and analysis based on convolutional neural net-work," IEEE Transactions on Semiconductor Manufacturing, vol. 32, no. 4, 2019, doi: 10.1109/TSM.2019.2937793.

[20] N. Yu, H. Chen, Q. Xu, M. M. Hasan, and O. Sie, "Wafer map defect patterns classification based on a lightweight network and data augmentation," CAAI Trans Intell Technol, 2022, doi: 10.1049/cit2.12126.

[21] R. Doss, J. Ramakrishnan, S. Kavitha, S. Ramkumar, G. Charlyn Pushpa Latha, and K. Ramaswamy, "Classifica-tion of Silicon (Si) Wafer Material Defects in Semiconductor Choosers using a Deep Learning ShuffleNet-v2-CNN Model," Advances in Materials Science and Engineering, vol. 2022, 2022, doi: 10.1155/2022/1829792.

[22] C. Phua and L. B. Theng, "Semiconductor wafer surface: Automatic defect classification with deep CNN," in IEEE Region 10 Annual International Conference, Proceedings/TENCON, 2020. doi: 10.1109/TENCON50793.2020.9293715.

[23] J. Yan, Y. Sheng, and M. Piao, "Semantic Segmentation Based Wafer Map Mixed-Type Defect Pattern Recogni-tion," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2023, doi: 10.1109/TCAD.2023.3274958.

[24] 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.

[25] C. Suresh, S. Singh, R. Saini, and A. K. Saini, "A Comparative Analysis of Image Scaling Algorithms," Interna-tional Journal of Image, Graphics and Signal Processing, vol. 5, no. 5, 2013, doi: 10.5815/ijigsp.2013.05.07.

[26] D. D. Muresan and T. W. Parks, "Adaptively Quadratic (AQua) image interpolation," IEEE Transactions on Image Processing, vol. 13, no. 5, 2004, doi: 10.1109/TIP.2004.826097.

[27] A. Bansal, R. Sharma, and M. Kathuria, "A Systematic Review on Data Scarcity Problem in Deep Learning: Solu-tion and Applications," ACM Comput Surv, vol. 54, no. 10, 2022, doi: 10.1145/3502287.

[28] D. Hirahara, E. Takaya, T. Takahara, and T. Ueda, "Effects of data count and image scaling on Deep Learning training," PeerJ Comput Sci, vol. 6, 2020, doi: 10.7717/peerj-cs.312.

[29] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning Augmentation Policies from Data," Cvpr 2019, no. Section 3, 2018.

[30] S. Wang, Z. Zhong, Y. Zhao, and L. Zuo, "A Variational Autoencoder Enhanced Deep Learning Model for Wafer Defect Imbalanced Classification," IEEE Trans Compon Packaging Manuf Technol, vol. 11, no. 12, 2021, doi: 10.1109/TCPMT.2021.3126083.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun ACM, vol. 60, no. 6, 2017, doi: 10.1145/3065386.

[32] C. Seger, "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing," Degree Project Technology, 2018.

[33] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems, 2019.

[34] S. Narang et al., "Mixed precision training," in 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings, 2018.

[35] PyTorch, "Datasets & DataLoaders." Accessed: Nov. 17, 2023. [Online]. Available: https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, 2014.

[37] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolu-tional neural networks for image classification," in IJCAI International Joint Conference on Artificial Intelligence, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-210.

[38] V. Romanuke, "Appropriate Number and Allocation of RELUS in Convolutional Neural Networks," Research Bulletin of the National Technical University of Ukraine "Kyiv Politechnic Institute," vol. 0, no. 1, 2017, doi: 10.20535/1810-0546.2017.1.88156.

[39] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural net-works," Artif Intell Rev, vol. 55, no. 1, 2022, doi: 10.1007/s10462-021-10033-z.

[40] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," Lecture Notes in Com-puter Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7700 LECTURE NO, 2012, doi: 10.1007/978-3-642-35289-8_26.

[41] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimiza-tion Framework," in Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019. doi: 10.1145/3292500.3330701.

[42] E. Brochu, T. Brochu, and N. Freitas, "A Bayesian interactive optimization approach to procedural animation design," in Computer Animation 2010 - ACM SIGGRAPH / Eurographics Symposium Proceedings, SCA 2010, 2010.

[43] H. Jiang and X. Li, "Parameter estimation of statistical models using convex optimization," in IEEE Signal Pro-cessing Magazine, 2010. doi: 10.1109/MSP.2010.936018.

[44] A. D. Jagtap, K. Kawaguchi, and G. Em Karniadakis, "Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks," Proceedings of the Royal Society A: Mathematical, Physical and En-gineering Sciences, vol. 476, no. 2239, 2020, doi: 10.1098/rspa.2020.0334.

[45] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in 7th International Conference on Learning Representations, ICLR 2019, 2019.

[46] I. Loshchilov and F. Hutter, "Fixing Weight Decay Regularization in Adam," 2018.

[47] D. Silwal, "Confusion Matrix, Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures," Linkedin.

[48] R. Joshi, "Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures - Exsilio Blog," 2016.

[49] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," Radiology, vol. 143, no. 1, 1982, doi: 10.1148/radiology.143.1.7063747.

[50] C. Marzban, "The ROC curve and the area under it as performance measures," Weather Forecast, vol. 19, no. 6, 2004, doi: 10.1175/825.1.

[51] Y. Bai, Q. Liu, W. Wu, and Y. Feng, "cuSCNN: A Secure and Batch-Processing Framework for Privacy-Preserving Convolutional Neural Network Prediction on GPU," Front Comput Neurosci, vol. 15, 2021, doi: 10.3389/fncom.2021.799977.

[52] G. V. Stoica, R. Dogaru, and E. C. Stoica, "Efficient image processing using reaction-diffusion CNN implemented in CUDA technology," UPB Scientific Bulletin, Series C: Electrical Engineering and Computer Science, vol. 78, no. 2, 2016.

[53] Programming Massively Parallel Processors. 2023. doi: 10.1016/c2020-0-02969-5.