Semiconductor defect classification by using cascaded convolutional neural network

P N A Tajudin1* and M I Shapiai1
1Center of Artificial Intelligent and Robotics, Malaysia-Japan International Institute of Technology, Universiti Teknologi Malaysia, 54100 Kuala Lumpur, Malaysia.

*Corresponding author: putrinorazila21@gmail.com

Abstract. The semiconductor industry is one of the rapidly growing industries globally. To be on par in both quality and quantity, most of the semiconductor industry has an inspection procedure in line with every production process. The dataset defect classes' main features can be group into two main groups base on their pattern similarity. Previously, most of the studies did not consider the pattern similarity and fed all the input into one classifier, increasing the risk of misclassification into more classes. One classifier does not feed all. Therefore, this study proposed a cascaded convolutional neural network classifier framework for semiconductor defects to enhance the defect pattern analysis. The cascaded convolutional neural network consists of multiple models trained based on various pre-process inputs. The data pre-processing includes a median filter method by using the OpenCV library. The input has to be pre-processed to distinguish the main feature, and the model can classify it based on those features. Comparing the proposed method with the single classifier show a slight increase in accuracy from 86.7% to 87.7%. In the future, further investigation will be carried out, especially on certain defect wafer compromise of multiple defect classes and how does it reduce the capability of the cascaded convolutional neural network model.

1. Introduction
The semiconductor industry is one of the booming industries that overgrow from over $1 billion in 1964 to $335.8 billion in 2014. In 2017, semiconductors worldwide sales exceeded the US $400 billion for the first time[1]. Higher demand impacts the manufacturing process, and to maintain yield, the manufacturer opted for the inspection system throughout the production floor process. Silicon wafer is a primary component of the majority of semiconductor devices. Silicon wafer undergoes numerous fabrication processes that commonly fall into four general processes: deposition, removal, patterning, and modification of electrical properties[2]. Throughout this process, the wafer will undergo multiple inspection processes depending on the companies standard operating procedure. A 100 percent yield is impossible because it will go through various methods, whether it is a fully automated process or even a process involving human workers. The inspection process maintains yield by informing the engineer's abnormalities for rapid maintenance response; it also maintains product quality to the consumer.

Up until recently, wafer defects were analyzed by trained process engineers using high-resolution microscopes by monitoring process criteria of the wafer map defect, such as location, size, and color. These abnormalities are categorized into Type A, Type B, and Type C[3]. Type A defect is considered as evenly random with stable mean density. This type of defect has generated randomly; no particular clustering pattern is perceptible, as shown in Figure 1(a). Type B defect is considered a systematic and repeatable defect, as shown in Figure 1(b). This type of defect has a perceptible
clustering pattern and usually caused by the production process. Type C defect is a combination of Type A and Type B defect shown in figure 1(c).

![Figure 1](image)

**Figure 1.** Wafer Defect Type (a) A/Random, (b) B/Systematic, and (c) C/Combination.

A public dataset of the semiconductor wafer is hard to come by, primarily to obtain the defect type that might tarnish the quality of the factory processes. However, there is a silicon wafer dataset provide by Multimedia Information Retrieval (MIR) Laboratory, which provides the WM-811K silicon wafer dataset for public use. Collected data came from real semiconductor industries. This particular dataset contains nine defect classes that the process engineer has uniquely identified. The defect class name is Center, Donut, Local, Edge-Local, Edge-Ring, Random, Scratch, Near-All, and None[4].

On this particular set of the dataset, there have been various studies that employ the usage of artificial intelligence, such as Support Vector Machine(SVM)[4], Artificial Neural Network (ANN)[5], Generative Adversarial Network (GAN)[6,7], and Convolutional Neural Network (CNN). SVM and ANN are a part of machine learning, and these kinds of methods required a lot of feature extraction that required manual intervention. On the other hand, CNN is a part of deep learning, and it does not require much manual intervention[8]. GAN is one of the methods introduced in deep learning, but it is costly due to the complex computational operation. Maksim et al. [9] proposed using a single Resnet50 for defect classification using only six classes out of nine defect classes. Batool et al.[10] propose a similar method but by using more basic CNN architecture. However, those research disregard some of the classes, and even with discarding some classes, the average accuracy of Maksim et al. paper is less than our proposed method.

This study proposed a cascaded convolutional neural network classifier framework for semiconductor defect problems to enhance the defect pattern analysis. The defect classes' main features can be categorized into two major groups based on their pattern similarity. The cascaded convolutional neural network consists of multiple classifiers trained based on different pre-process inputs that enhance the defect's primary feature. These studies also include all nine defect classes.

2. Research Methodology
This segment will briefly present the dataset, demonstrate the augmentation technique, and depict the CNN model's essential structure.

2.1. Dataset
WM-811K is a public dataset provided by MIR Lab. It contains 811 457 wafer map fabrication data collected in the actual fabrication process[4]. There are approximately 20% of the labeled dataset, and most of it comes from the "None" pattern class. The dataset contains nine defect types. The utilization of the label "None" class is just a portion of the entire label "None" class.

On the other hand, other classes are utilized fully. The distribution for training: validation: testing is 6:1.5:2.5. Figure 2 shows the distribution of data.
2.2. Data Pre-processing

The WM-811K wafer data contain different sizes of the die from 6×21 to 300×202. Die size of 24x24 and 224x224 selected for the classifier training and testing. The dataset was then filtered out to remove out the noise without removing the defect's primary characteristic. This experiment utilizes a median filter that supports by OpenCV. The median filter replaced the current location's pixel value with the median of all neighboring pixels[11]. The kernel for the median filter varies between the classifier. Classifier two received median filter input with kernel size five, while classifier three received median filter input with kernel size 3. The random rotation transformation method was applied to the input before feeding the three classifiers' input for training. Random rotation of a maximum of 180 degrees was applied using an "on the fly" augmentation technique.

2.3. Proposed Cascaded Classifier

The proposed classifier contains three separate classifiers, as shown in figure 3. Classifier A acts as a first filter that differentiates the input from classifier B or classifier C. The result of classifier one will send a file name either to classifier two or classifier three. Classifier B and classifier C will then retrieve input from their respective median filter kernel size folder. Input class that goes through classifier B are grouped due to the similarities of their defect pattern. Those defects pattern mainly have higher noise to remaining classes. Thus, it needs a higher value kernel size to smooth out the input images.

![Dataset Distribution](image)

**Figure 2.** Dataset Distribution.

| Class    | Testing | Validation | Training |
|----------|---------|------------|----------|
| Edge-Local | 1290    | 3113       | 2402     |
| Edge-Ring  | 492     | 5808       | 1452     |
| Scratch    | 215     | 215        | 778      |
| Random     | 132     | 319        | 215      |
| Near-Full  | 38      | 38         | 89       |
| Local      | 898     | 2155       | 538      |
| Donut      | 644     | 7500       | 1061     |
| Center     | 2576    | 18000      | 519      |
| None       | 715     | 299        | 898      |
|           | 89      | 178        | 538      |
|           | 178     | 2402       | 129      |
|           | 129     | 2155       | 22       |
|           | 22      | 1452       | 299      |
|           | 299     | 319        | 896      |
|           | 896     | 519        | 178      |
|           | 178     | 4500       | 129      |
|           | 129     | 7500       | 22       |
|           | 22      | 299        | 896      |
|           | 896     | 1452       | 178      |
|           | 178     | 4500       | 299      |
|           | 299     | 7500       | 129      |
|           | 129     | 519        | 896      |
|           | 896     | 7500       | 178      |
|           | 178     | 4500       | 299      |

In contrast, the other four classes, Donut, Center, Random, and Scratch, will go through classifier C. Classifier B are grouped due to the similarities of their defect pattern. Those defects pattern mainly have higher noise to remaining classes. Thus, it needs a higher value kernel size to smooth out the input images.
2.4 Convolutional Neural Network

Classifier A employs VGG-16 architecture. VGG-16 is an architecture that contains thirteen convolutional layers and three fully connected layers with a Rectified Linear Unit (ReLU) activation. Along with these, there are five max-pooling layers implemented between the convolutional layers. The kernel for the convolutional layer is three, and the input size is 224x224 [12].

The classifier B and C are a custom CNN, consisting of a combination of 2 convolutional layers, two max-pooling layers, and one dropout layer followed by a two-dense layer and the output layer at the end. The kernel size for the convolutional layer is five, while for the max-pooling layer and the kernel size is two. Classifiers B and C received input images of size 24x24.

The optimizer for classifier A and B is SGD, while for classifier C the optimizer is Adam. The different optimizer types because during the hyperparameter optimization, the respective optimizer works well with the respective classifier. Table 1 shows the network layer of the classifier and tabulated hyperparameter are in table 2. Visualization of the network configuration is in figure 4.

Table 1. The network layer of classifier B and classifier C.

| Layer  | Type       | Feature Map | Output Size | Activation |
|--------|------------|-------------|-------------|------------|
| IN     | Input      | 3           | 24x24       |            |
| C1     | Convolutional | 16          | 20x20       | ReLU       |
| P1     | Max-Pooling | 16          | 10x10       |            |
| C2     | Convolutional | 32          | 6x6         | ReLU       |
| P2     | Max-Pooling | 32          | 3x3         |            |
| L1     | Fully-Connected | 256        |             | ReLU       |
| L2     | Fully-Connected | 84         |             | ReLU       |
| OUT    | Output     | 4@5         | -           | LogSoftmax |
Figure 4. Convolutional neural network for classifier B and classifier C.

Table 2. Training hyperparameter for all three networks.

| Parameter         | Classifier A | Classifier B | Classifier C |
|-------------------|--------------|--------------|--------------|
| Optimizer         | SGD          | SGD          | Adam         |
| Learning rate     | 0.001        | 0.02         | 0.0001       |
| Weight decay      | 6e⁻⁵         | 5e⁻⁵         | 5e⁻³         |
| Transfer learning | Yes          | No           | No           |
| Epoch             | 100          | 100          | 100          |

3. Result and Discussion
This section will briefly present the performance, result, and discussion of the proposed cascaded CNN.

3.1. Performance
The batch size for the training is 32, and the number of epochs is set to 100 epochs for each classifier. During the training phase, all classifiers' input is randomly rotated with the "on-the-fly" augmentation method support by the Pytorch library. Training loss for classifier A, classifier B, and classifier C is approximately 0.025, 0.1, and 0.1, respectively. Validation loss for all classifiers is approaching zero. Compared with training loss, the lower validation loss is due to the augmentation technique used during the training phase, leading to validation data being much easier to predict. Besides that, all classifiers contain a dropout layer, which only activates during the training phase. The summarization of the losses shown in figure 5.
(a) Classifier A

(b) Classifier B
Figure 5. Losses (a) Classifier A, (b) Classifier B and, (c) Classifier C.

The confusion matrix in figure 6 demonstrates the model's prediction accuracy on the test data in terms of the number of samples and the percentage. The model test a total of 13835 images, which comprise 25% of the label class. It can be seen from the confusion matrix that None, Center, Donut, and Random class perform well with accuracy above 90%. The misclassification between Donut and Random class is not high even though those two classes are among the lowest percentage of data in terms of the total data use for training. The reason is that those two classes have a very distinguishable pattern defect. Edge-Local and Edge-Ring show the worst performance, which got confused with classes within the B classifier grouping. The reason is due to the similarity between the defect pattern in grouping B itself.

The cascaded classifier shows that it can correctly handle the None class base on the performance matric in table 3. This is important to avoid false alarms on the production floor. The Donut, Local, and Random class performance matric show that the class is well detected, but the classifier does include another class point.

Table 3. Cascaded classifier performance matric.

| Class       | Precision | Recall | F1-Score |
|-------------|-----------|--------|----------|
| Center      | 0.96      | 0.91   | 0.94     |
| Donut       | 0.81      | 0.94   | 0.87     |
| Edge-Local  | 0.71      | 0.78   | 0.74     |
| Edge-Ring   | 0.99      | 0.79   | 0.88     |
| Local       | 0.80      | 0.89   | 0.84     |
| Random      | 0.74      | 0.96   | 0.83     |
| Scratch     | 0.79      | 0.82   | 0.81     |
| Near-Full   | 0.70      | 0.82   | 0.76     |
| None        | 0.95      | 0.97   | 0.96     |
3.2. Comparison Between Cascaded CNN And Single VGG-16

The comparison between using a single VGG-16 and the proposed classifier achieves a slight increase in the average accuracy. Overall test accuracy of 87.7% against 86.7% of single VGG-16 has been observed. Both Edge-Local and Edge-Ring have the most badly predicted class by VGG-16 show a slight increase by the proposed classifier. This justifies the proposed method to categories the group into two main groups.

The primary characteristic of group B classes can hardly be distinguished with naked eyes such as Edge-Local, None, and Local. The proposed method tries to eliminate the risk of misclassification between group B and group C while emphasizing classifier B to learn to distinguish the features of four classes in group B more thrivingly.

In the case of classifier C, the worst performed class is Near-Full compared with single VGG-16. Near-Full class misclassification with Random class caused by the median filter process that erases the Near-Full class's primary feature. Overall, the classifier shows an increase in accuracy for six classes while underperforming in three other classes. A comparison of the testing accuracy results is in table 4.

### Table 4. Comparison of testing accuracy.

| Class      | Cascaded Classifier | Single-VGG-16 |
|------------|---------------------|---------------|
| Center     | 90.9                | 81.4          |
| Donut      | 94.2                | 94.02         |
| Edge-Local | 77.9                | 77.6          |
| Edge-Ring  | 79.4                | 77.8          |
| Local      | 89.2                | 89.8          |
| Random     | 96.3                | 89.8          |
| Scratch    | 82.3                | 81.3          |
| Near-Full  | 81.6                | 89.5          |
| None       | 97.2                | 98.6          |
| **Average** | **87.7**            | **86.7**      |

The simulation of this experiment utilizes Nvidia GeForce GTX 1080 GPU. The time taken to evaluate one image is 9.4ms for a cascaded classifier and 6.8ms for a single VGG-19. The cascaded
classifier needs a slightly higher time due to three classifiers than the VGG-16 that only uses one classifier.

4. Conclusion
This study proposed a cascaded convolutional neural network for defect classification that thoroughly analyses the defect's characteristics. The comparison between using a single VGG-16 and the proposed classifier achieves a slight increase in the average accuracy. The proposed network show an average test accuracy of 87.7% against 87.7% of single VGG-16. Even though the accuracy did not significantly increase, it shows that with more proper characteristic analysis and much more enhance CNN for classifier B and C, the proposed method can improve significantly. The cascaded classifier shows the more practical use of CNN in solving real problems because raw data is usually corrupt with noises, and these noises vary between classes. The usage of multiple classifiers can distinguish the characteristics more thoroughly. There is a problem with the imbalance of data training with the "on the fly" augmentation methods. In the future, further investigation will be carried out, especially on certain defect wafer that compromises of multiple defect classes, and how does it reduce the capability of the cascaded convolutional network model.

Acknowledgment
The authors would like to thank Universiti Teknologi Malaysia for Industrial Defect Detection for Wafer Semiconductor using Artificial Intelligence Technology grant (Q.K130000.3043.01M84) and Imbalanced Strategy for Wafer Defect Classification Using Fully Convolutional Neural Network grant (Q.K130000.2543.21H17).

References
[1] Semiconductor Industry Association Annual Semiconductor Sales Increase 21.6 Percent, Top $400 Billion for First Time - Semiconductor Industry Association.
[2] Bera B 2019 Silicon Wafer Cleaning: A Fundamental and Critical Step in Semiconductor Fabrication Process Int. J. Appl. Nanotechnol. 8–13.
[3] Chien J C, Wu M T and Lee J Der 2020 Inspection and classification of semiconductor wafer surface defects using CNN deep learning networks Appl. Sci. 10 1–13.
[4] Wu M J, Jang J S R and Chen J L 2015 Wafer map failure pattern recognition and similarity ranking for large-scale data sets IEEE Trans. Semicond. Manuf. 28 1–12.
[5] Saqlain M, Jargalsaikhan B and Lee J Y 2019 A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing IEEE Trans. Semicond. Manuf. 32 171–82.
[6] Ring E Using GAN to Improve CNN Performance of Wafer Map Defect Type Classification.
[7] Wang J, Yang Z, Zhang J, Zhang Q and Chien W T K 2019 AdaBalGAN: An Improved Generative Adversarial Network with Imbalanced Learning for Wafer Defective Pattern Recognition IEEE Trans. Semicond. Manuf. 32 310–9.
[8] Schmidhuber J 2015 Deep Learning in neural networks: An overview Neural Networks 61 85–117.
[9] Maksim K, Kirill B, Eduard Z, Nikita G, Aleksandr B, Arina L, Vladislav S, Dаниil M and Nikolay K 2019 Classification of wafer maps defect based on deep learning methods with small amount of data 2019 Int. Conf. Eng. Telecommun. EnT 2019 1–5.
[10] Batool U, Shapiai M I, Fauzi H and Fong J X 2020 Convolutional Neural Network for Imbalanced Data Classification of Silicon Wafer Defects 2020 16th IEEE Int. Colloq. Signal Process. its Appl. CSPA 2020 (Langkawi, Malaysia) 230–5.
[11] Culjak I, Abram D, Pribanic T, Dzapo H and Cifrek M 2012 A brief introduction to OpenCV MIPRO 2012 - 35th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. (Croatia) 1725–30.
[12] Simonyan K and Zisserman A 2015 Very deep convolutional networks for large-scale image recognition 3rd Int. Conf. Learn. Represent. ICLR 2015 (San Diego, USA) 1–14.