A Deep Convolutional Neural Network for Wafer Defect Identification on an Imbalanced Dataset in Semiconductor Manufacturing Processes

Muhammad Saqlain, Qasim Abbas, and Jong Yun Lee

***Abstract*—Wafer maps contain information about various defect patterns on the wafer surface and automatic classification of these defects plays a vital role to find their root causes. Semiconductor engineers apply various methods for wafer defect classification such as manual visual inspection or machine learning-based algorithms by manually extracting useful features. However, these methods are unreliable, and their classification performance is also poor. Therefore, this paper proposes a deep learning-based convolutional neural network for automatic wafer defect identification (CNN-WDI). We applied a data augmentation technique to overcome the class-imbalance issue. The proposed model uses convolution layers to extract valuable features instead of manual feature extraction. Moreover, state-of-the-art regularization methods such as batch normalization and spatial dropout are used to improve the classification performance of the CNN-WDI model. The experimental results comparison using a real wafer dataset shows that our model outperformed all previously proposed machine learning-based wafer defect classification models. The average classification accuracy of the CNN-WDI model with nine different wafer map defects is 96.2%, which is an increment of 6.4% from the last highest average accuracy using the same dataset.**

***Index Terms*—Wafer maps, wafer defect identification, deep learning, convolutional neural network, data augmentation, batch normalization.**

I. Introduction

SEMICONDUCTOR manufacturing processes should produce high-quality products and improve the wafer yield to fulfill market demands. However, semiconductor fabrication is a very complex, costly, and time-consuming process that involves various chemical, mechanical, and electrical processes such as deposition, etching, photolithography, chemical planarization, ion implantation, and diffusion [1]. After applying all these processes, integrated circuits (ICs) are composed by making circuit structures on various layers of the same wafer and joining them with wires. A wafer surface should be very clean, and all layers of the circuit should be perfectly aligned to produce high-quality ICs. However, well-trained semiconductor engineers working with highly automated and precise equipment in a very clean environment, cannot produce error-free wafer dies [2]. At the end of the fabrication process, each wafer goes through a circuit prob test where defective and defect-free wafer dies are differentiated and the results of the test are represented in wafer map [WM], which is a 2-dimensional (2D) wafer image.

Experienced process engineers are hired to define the wafer defect patterns and give them unique labels such as *Center*, *Donut*, *Local*, *Edge-Loc*, *Edge-Ring*, *Scratch*, *Random*, *Near-Full*, and *None* [3]. Additionally, these defects become more common due to increasing integration density of circuits and the wafer design complexity. Each wafer defect occurs due to specific abnormal behavior of some fabrication process. For instance, *Center* defects may occur because of uniformity issues in chemical and mechanical planarization, *Edge-Loc* defects may occur because of thin film deposition, and *Edge-Ring* defects occur due to etching problems [4]. So, WM defect analysis provides crucial information to discover the abnormal processes in semiconductor manufacturing and to take measures to resolve them.

Accurate classification of WM patterns plays an important role in identification of wafer defects, which will enhance the semiconductor yield and quality by improving the wafer fabrication process. Previously, wafer defects were examined by experienced process engineers using high-resolution microscopes by measuring the physical parameters of the WMs like location, size, and color. Moreover, various machine learning (ML) based automated defect classification (ADC) systems were introduced to reduce labor and fabrication costs while improving quality and yield. Wu *et al.* [3] proposed a model called WMFPR, which extracts Radon-based and geometry-based features from WM and applies a support vector machine (SVM) classifier to classify wafer defect patterns. Piao *et al.* [5] extracted Radon transform features and applied an ensemble-based decision tree model to classify various WM defects. Recently, Saqlain *et al.* [6] also proposed an ensemble-based WM classification model called WMDPI, which combines state-of-the-art ML classifiers such as random forest (RF), logistic regression (LR), SVM, and artificial neural network (ANN). Three different types of features were

Manuscript received November 26, 2019; revised March 10, 2020; accepted May 9, 2020. Date of publication May 14, 2020; date of current version August 4, 2020. This work was supported in part by the Korea Institute for Advancement of Technology Grant Funded by the Korea Government (Ministry of Trade Industry and Energy) under Grant N0002429, as well as was Collaborated with SIMTech in Singapore, and in part by the Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education under Grant 2017R1D1A1A02018718. *(Corresponding author: Jong Yun Lee.)*

The authors are with the Department of Computer Science, Chungbuk National University, Cheongju 28644, South Korea (e-mail: jongyun@chungbuk.ac.kr).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSM.2020.2994357

extracted from the WMs like Radon-based, geometry-based, and density-based, but the classification accuracy of their model was very poor for some specific defect classes and a lot of manual inspections like features extractions and hyperparameter setting were also needed.

Through previous studies, it can be found that most of the WM defect classification models require manually extracted useful features. To do so, experienced semiconductor engineers first analyze the wafer surface and understand the physical measurements, and then propose valuable features according to the corresponding diagnosis issue. Such models are relatively costly, time-consuming, and inefficient in the presence of big WM datasets. However, a deep learning (DL) based classifier such as convolutional neural networks (CNNs) does not require manually extracted features for classification [7]. The CNN consists of three types of layers such as convolution layers, pooling layers, and fully connected layers. Whereas the convolution layer is typically used to extract features, the pooling layer summarizes the extracted features by reducing the size, and fully connected layers finally classify the input image using the extracted features [8].

Recently, many studies have been conducted that used CNN models to classify wafer defect patterns from the original WM images [9]. For example, Nakazawa and Kulkarni [10] applied CNN with a softmax activation function in the final layer to classify 22 WM defect patterns. The dataset they used was very small and highly imbalanced, so only simulated data was used to train and validate the model. Kyeong and Kim [11] also proposed a CNN model to classify single as well as mixed defect patterns on the same WM. They developed multiple CNN models and each model classifies a specific defect class which was practically very expensive in sense of time and computation. Cheon *et al.* [1] proposed a CNN model that can extract features from the real wafer image and accurately classified the input data into five different wafer defect classes. Their model can also classify the unknown defect classes after combining the CNN model with the k-nearest neighbors ($k - NN$) algorithm. The datasets used by all these studies were very small and highly imbalanced. Whereas, CNN models can get higher training accuracy in the presence of bigger datasets [12]. Additionally, imbalanced data distribution of various classes may cause the CNN to be biased for the majority data sample classes [13]. None of the previous studies has defined a suitable solution to overcome the data imbalanced issue for WM classification.

In this paper, we propose a deep layered CNN-based wafer defect identification (CNN-WDI) model in a semiconductor manufacturing process. We use a real wafer dataset called WM-811K, which consists of nine different labeled classes of WM defect patterns. All defect classes are equally important because each of these occurs due to some specific abnormal behavior of the fabrication process. Semiconductor engineers classify the wafer defects to find the abnormal behavior behind these defects. But the available dataset is highly imbalanced that may ignore the classification of minority data sample classes. Thus, we implement a data augmentation method to increase the size of minority defect classes. Finally, we apply the proposed CNN-WDI model on the balanced dataset of 9 defect classes. For example, batch normalization and spatial dropout methods have been applied for regularization of the model. The experimental results comparison shows that CNN-WDI model has outperformed all previous models in terms of classification accuracy. The average classification accuracy of the proposed model with nine different WM defect patterns is 96.2%, which is an increase of 6.4% from the last highest average accuracy using the same dataset.

The remainder of this research study is organized as follows. Section II introduces methods and material and Section III describes the proposed CNN-WDI model. Section IV presents experimental results and performance evaluation. Finally, Section V summarizes the whole study and gives a plan for our future research study.

II. Methodology

It is necessary to have a basic knowledge of CNN operations to understand the phenomenon of automatic feature extraction from WMs. This section briefly introduces the dataset, illustrates the data augmentation method, and describes the basic structure of CNN model.

A. Dataset

The WM-811K dataset is a semiconductor dataset which consists of 811,457 real WM images [3]. The wafer images were collected from 46,293 lots in a circuit probe (CP) test of semiconductor fabrication process. A single lot contains 25 WMs, so there should be 1,157,325 WMs in total (i.e., 46,293 lots $\times$ 25 wafer/lot). Since not all lots have exact 25 WMs due to some sensor faults or other unknown reasons, they were pruned from the dataset. The dataset also contains additional information about each WM such as lot name, die size, wafer index number, failure types, and training/test labels. This is the largest publicly available WM dataset that can be accessed at the MIR laboratory website [14]. There are different sizes of wafer images because of their two-dimensional nature and having different pixel values along the length and width of the image. We found total of 632 various sizes of wafer images ranging from (6×21) to (300×202).

Domain experts were responsible for defining nine different defect classes of WMs and assigning manual labels to 172,950 (21.3%) WMs in whole dataset. Unfortunately, the labeled dataset is highly imbalanced and only *None* defect class occupied 147,431 (85.2%) WMs of labeled dataset. The other eight defect classes that contains 25,519 (14.8%) WMs of labeled dataset as total are given as *Center:* 4294 (2.5%), *Donut:* 555(0.3%), *Edge-Loc:* 5189 (3.0%), *Edge-Ring:* 9680 (5.6%), *Local:* 3593 (2.1%), *Random:* 866 (0.5%), *Scratch:* 1193 (0.7%), and *Near-full:* 149 (0.1%). Fig. 1 shows the randomly selected wafer defect images from each class.

B. Data Preprocessing

Wafer defect images were sampled from real-world dataset WM-811K which consists of 811,457 original WMs but only 21% of these wafers contain labeled classes. This dataset is highly imbalanced which leads to overtraining the majority data sample classes during the training of CNN-WDI model.

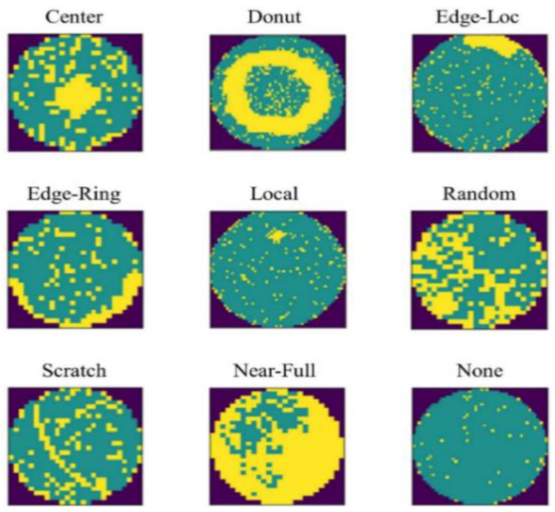

Fig. 1. Typical examples of nine wafer defect classes.

Thus, we applied data augmentation using random rotation of 10-degree, horizontal flipping, and width shift, height shift, shearing range, channel shifting, and zooming by 20%, 20%, 15%,10%, and 10%, respectively. Thus, we increased the size of minority data sample classes up to 133%, 1702%, 93%, 178%, 1055%, 738%, and 6611% from the original data of *Center*, *Donut*, *Edge-Loc*, *Local*, *Random*, *Scratch*, and *Near-Full* classes, respectively. So, our new dataset consists of 90,000 wafer defect images of nine different balanced classes with 10,000 data samples of each class. The dataset was distributed into training, validation, and testing subsets at proportions of 65%, 20%, and 15%, respectively.

C. Data Augmentation

Our dataset is not equally distributed and some of the wafer defect classes contain abundant data, while other classes have very little data. This issue is known as *class-imbalance problem*, and most image databases face this issue [15]. This unequal distribution of wafer defect classes can force the classification model to get higher accuracy for majority data sample classes during their training phase. Consequently, the training accuracy of the minority data sample class will be lower. The class-imbalance problem may also result in overfitting of training algorithm that means the algorithm will get higher training accuracy but have low testing performance. As all the wafer defects are equally important, it is necessary to correctly identify all wafer defect classes including minority data sample classes.

The easiest method to handle the class-imbalance and overfitting issues is to artificially increase the dataset of minority data sample classes. This method is called *data augmentation* and its commonly being used as model regularization technique in recent studies [8], [16], [17]. Deep neural networks (DNNs) can be trained well on massive amount of image datasets due to recent increasing computation capabilities. Thus, data augmentation becomes an effective method to improve both diversity and size of dataset by randomly augmenting the original data. Some common augmentation methods include flipping the image vertically or horizontally, shifting the image vertically or horizontally, and slightly rotating or zooming the image. This method helps the training model to be highly tolerant to changes in the size, position, and orientation of defects in the pictures.

D. Convolutional Neural Network (CNN)

Recently neural networks, especially multilayer perceptron (MLP) and DNNs which have multilayered neural network structures got huge attention in the field of machine learning [18]. The DNN can achieve better performance than the traditional neural networks because it has more hidden layers which alternatively means more learning ability. One of the most popular DNN models is convolutional neural network (CNN). CNN is an advanced form of MLP and specially designed to classify the images [19]. It has many advantages over traditional ML classifiers because of its similarity to the human visual cortex and the ability to extract and learn 2-dimensional (2D) features. Moreover, the number of parameters of CNN are very few as compared to the same size of DNN architecture. Most CNN models are trained using a gradient-based learning approach to overcome the vanishing gradient problem (VGP). The VGP badly affects the lower layers of CNN and makes the training process very hard [20].

The basic architecture of CNN model consists of two major parts: a feature extraction network and a classification network as shown in Fig. 2. The original image enters the feature extraction network, where it passes through sequential pairs of convolutional (Conv) and pooling (Pool) layers for extraction of useful features. These features are used by the classification network to classify the input image. Each hidden layer of the CNN takes the output of the previous layer as its input and after processing, forwards its output to the next layer as the input. Both convolution and pooling layers create a group of 2D planes called feature maps. Higher-level features are extracted from features generated by lower layers. Various sizes of kernels are used in convolution layer to reduce the dimension of extracted features. The output of the last layer of the feature extraction network is used as the input of the classification network. The classification network contains a fully connected feed-forward neural network because of its better performance [21]. But these layers are comparatively expensive in terms of network parameters because each node of every layer is fully connected with each other. The final classification result is calculated by output layer or Softmax layer by giving the maximum values to the accurate classes. More details about each layer of CNN model can be found in [22].

III. Proposed Deep CNN Model

A supervised learning approach is applied in the presence of labeled dataset. The WM-811K is a labeled dataset with 9 defect classes and each class contains complex features or patterns according to the defect nature. In this paper we constructed a 2D CNN model for WDI with one input layer, eight Conv layers each with Batch normalization (BN), padding, and Rectified Linear Unit (ReLU) activation, five Pool layers (four stacking pairs of Conv-Pool-Conv), one dropout layer, two

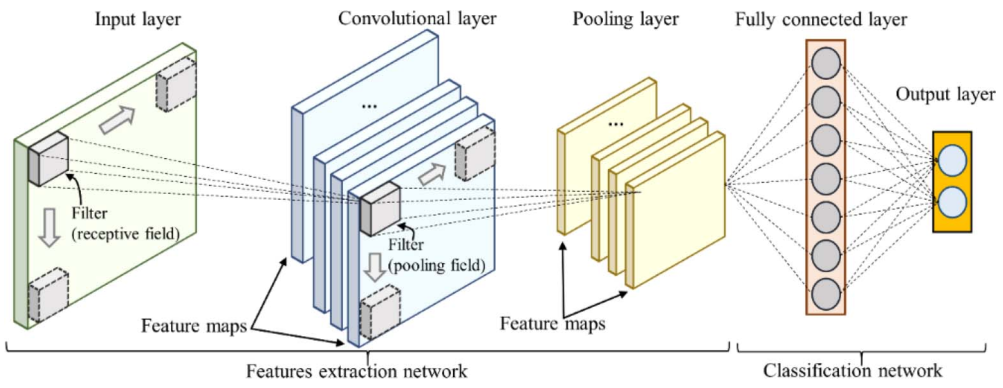

Fig. 2. The overall architecture of a convolutional neural network.

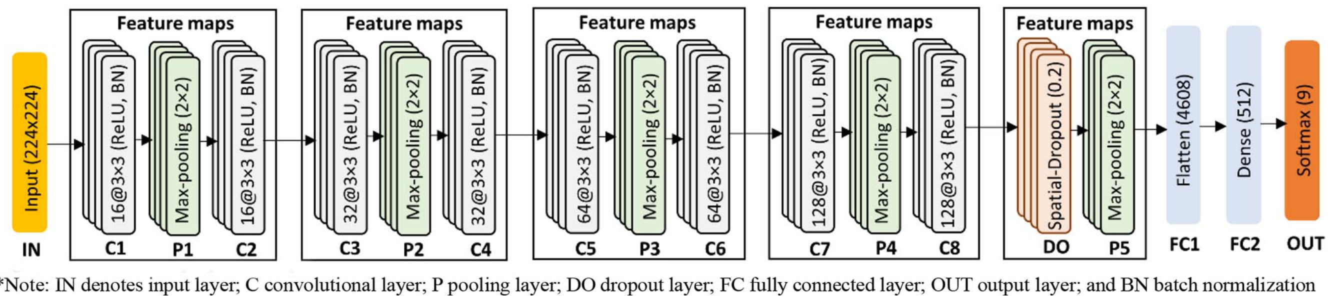

Fig. 3. Architecture of proposed deep CNN model for wafer defect identification.

fully connect (FC) layers, and one output layer. The layout of the proposed CNN model for WDI is shown in Fig. 3.

The first Conv layer extracts the features from input training wafer images of size (224×224). Each Conv layer contains a set of learnable filters to extract unique feature maps. The number of filters increases with increasing depth of the Conv layer, so the number of feature maps also increases. However, feature maps become smaller and complex due to the Pool layer in deeper network. The proposed CNN-WDI model adopts 16, 32, 64, and 128 feature maps for the first, second, third, and fourth stacking pairs, respectively. As the number of stacking pairs increases, two results are induced: 1) the ability of CNN model to extract more distinct features, which is significant for WDI, is increased, and 2) the chance of loss of information due to small feature map size is reduced.

In this model, each Conv and Pool layer consists of subsampling filters of size 3×3 and 2×2, respectively. The small filter size assures the extraction of detailed features. The ReLU activation function was applied in all layers except Pool and output layers. The ReLU was proposed to solve the VGP during training process of deep learning models [8]. It simply keeps all the values above zero unchanged and replaces all negative values with zero. Although, the ReLU function significantly reduces the VGP at the start of training, but it doesn't guarantee that the same problem will not come back during further training. So, we used BN operation to address the VGP throughout the whole training process of the CNN model. The BN operation was designed to reduce the shift of internal covariance of DNNs. In other words, it lets the CNN model learn the optimal mean and scale of each input of the Conv layer by normalizing and zero-centering. The more detail of the BN method can be found in [23].

For regularization of the CNN model, *Dropout* method is used which is a very simple but effective approach to prevent overfitting [24]. It improves the generalization performance of the network by avoiding activations from becoming highly correlated, which in turn leads to overfitting [25]. The main idea is to randomly remove or drop-out neurons and their connections by zeroing the activation of these neurons during training with some probability p_{drop}. We applied an advanced form of dropout called *SpatialDropout* (SD), proposed by Tompson *et al.* [26]. The standard dropout method fails to get required results when the network is fully convolutional, and activations of the feature maps are strongly correlated. The SD dropped-out entire feature maps of size $n_f \times H \times W$ from the Conv layer which are then not used by pooling operation. Therefore, the pixels in the SD feature map are either all activated or all deactivated. Thus, SD helps to promote independence of feature maps of the Conv layer. We applied the SD function between the last Conv layer and last pooling layer with a rate of 0.2, that means 20% of the randomly selected nodes will be dropped-out for each weight update cycle.

Zero padding was applied in all Conv layers to make sure that the dimensions of input and output feature maps are same. The Softmax activation function was applied to the output layer of our model. In addition, the Adam optimization method, which combines the concepts of Momentum optimization and root mean squared prop (RMSProp), was selected as the optimizer. This optimizer helps to achieve higher accuracy and improve training

TABLE I
PROPOSED DEEP CNN MODEL PARAMETERS

Layer	Type	Feature Maps	Output Size	Filter size	Padding	Activation
IN	Input	3 (RGB)	224×224	-	-	-
C1	Convolution1	16	222×222	3×3	No	ReLU
P1	Max Pooling1	16	111×111	2×2	No	-
C2	Convolution2	16	111×111	3×3	Yes	ReLU
C3	Convolution3	32	111×111	3×3	Yes	ReLU
P2	Max Pooling2	32	55×55	2×2	No	-
C4	Convolution4	32	55×55	3×3	Yes	ReLU
C5	Convolution5	64	55×55	3×3	Yes	ReLU
P3	Max Pooling3	64	27×27	2×2	No	-
C6	Convolution6	64	27×27	3×3	Yes	ReLU
C7	Convolution7	128	27×27	3×3	Yes	ReLU
P4	Max Pooling4	128	13×13	2×2	No	-
C8	Convolution8	128	13×13	3×3	Yes	ReLU
P5	Max Pooling5	128	6×6	2×2	No	-
FC1	Fully-Connected1	1	4608	-	-	ReLU
FC2	Fully Connected2	1	512	-	-	ReLU
OUT	Output	1	9	-	-	Softmax

process of the CNN model [27]. Besides these, after many attempts, some other parameters like batch size and the number of epochs were assigned 100 and 20, respectively. The smaller batch size improves the generalization ability of the CNN model by computing an approximation of the gradient value and then updating the other parameters [28]. The detailed parameters of CNN-WDI model are given in Table I.

IV. EXPERIMENTS AND DISCUSSION

In this section, we describe the training of the proposed CNN-WDI model, and its performance is compared with previously proposed models using various performance measures.

A. Training of CNN-WDI

The training data was used to train the parameters of proposed CNN-WDI model which helps to reduce the loss function. We applied the categorical cross-entropy as our loss function to measure the loss function between estimated output probability distribution and actual class probability distribution. Batch normalization and spatial dropout with the probability of 0.2 were used for the regularization of the model. During the training, we applied backpropagation algorithm to calculate the gradient of loss function. Moreover, Adam Stochastic optimizer was applied to minimize the loss function with the learning rate and batch size of 0.001 and 100, respectively. The validation dataset is used to evaluate the model by fine-tuning its various hyperparameters. We evaluated the classification accuracy of our trained CNN-WDI model using this dataset. A well-trained deep learning model not only gets higher classification accuracy for training data but also for validation data. When we have all done for our training the model, next we have checked that how accurately it performs with testing dataset. The testing dataset is totally unseen for the model and it provides an unbiased evaluation of the final CNN-WDI model on the training dataset.

The experiment was conducted on the personal computer with the following hardware specifications: Intel Xeon CPU E5-2696 v5 @ 4.40 GHz, 512 GB RAM, and NVIDIA GeForce GTX 1080 24 GB. The model was developed using *TensorFlow* [29] and *Keras* [30] libraries in *Jupyter Notebook* [31], which can handle different versions of Python language.

B. Performance Measures

The proposed CNN-WDI model is evaluated using different performance measures like accuracy, precision, recall, and F1-score. The accuracy of a classifier defines that how often it correctly predicts from whole data and defined as equation (1).

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

where TP, FP, FN, and TN show the true positive, false positive, false negative, and true negative values, respectively. Both precision and recall are objection functions of any ML classifier which are defined as follows.

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

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

Equations (2) and (3) show that both performance measures are inversely proportional to each other and each of them has various classification measuring qualities. While, F1-score calculates the harmonic mean of precision and recall, and it is defined as follows.

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{4}$$

Equation (4) shows that $F1 - score$ is the interpretation between actual and predicted probabilities. If these probabilities are close to each other than $F1 - score$ will show the higher result and vice versa.

TABLE II
PERFORMANCE EVALUATION OF CNN-WDI FOR BALANCED AND IMBLANCED DATASETS (%)

Defect Class	Balanced dataset			Imbalanced dataset		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Center	93.6	**98.0**	95.7	**94.7**	97.7	**96.2**
Donut	**96.0**	**98.3**	**97.2**	85.2	90.4	87.7
Edge-Loc	**97.4**	**93.1**	**95.2**	89.2	87.4	88.3
Edge-Ring	94.4	91.7	93.0	**97.7**	**98.3**	**98.0**
Local	**91.6**	**90.3**	**90.9**	82.8	81.4	82.1
Near-Full	98.9	**99.9**	**99.4**	**100.0**	59.1	74.2
Random	**96.7**	**96.5**	**96.6**	83.0	90.0	86.4
Scratch	**98.1**	**98.7**	**98.4**	80.4	73.2	76.6
None	99.5	99.7	99.6	**99.9**	**100.0**	**100.0**
Average	**96.24**	**96.24**	**96.22**	90.32	86.39	87.72

Note: Boldface numbers denote the highest values of different performance measures between balanced and imbalanced datasets

C. CNN-WDI Results and Analysis

Table II presents the classification results of the CNN-WDI model using the original imbalanced dataset and the augmented balanced dataset. It can be seen in Table II that the proposed model got a very high value of precision, recall, and F1-score for all defect pattern classes for a balanced dataset. But the same model performed very poorly for most of the minority classes in the presence of an imbalanced dataset. For instance, the values of F1-score of minority classes like *Near-Full*, *Scratch*, and *Local* defect classes are only 74.2%, 76.6%, and 82.1%, respectively, which are comparatively very low. Moreover, this model got the value of the FI-score of *None* class up to 100% because it is a majority class of imbalanced dataset. The average values of the precision, recall, and F1-score of the CNN-WDI model for the balanced dataset are 96.24%, 96.24%, and 96.22%, respectively. Whereas, the same value for the imbalanced dataset are 90.32%, 86.39%, and 87.72, respectively. Thus, the data augmentation method improves the performance measure values of precision, recall, and F1-score up to 6.6%, 11.4%, and 9.7%, respectively. The reason is simple, CNN model gets better learning in the presence of large datasets. Therefore, these results show that the balancing dataset through data augmentation method can play a vital role in WM defect identification.

We draw a confusion matrix to represent the number of accurately classified and misclassified defect patterns of all classes as shown in Fig. 4, where *predicted label* and *true label* along the *x-axis* and *y-axes*, respectively, represents labels of various defect classes. The *predicted label* shows the number of predicted defect patterns of a specific class and the *true label* shows the numbers of actual defect patterns of the corresponding defect class. The main diagonal values show the ratio of correctly classified defect patterns. As the *Local* defect is very similar to *Donut*, *Edge-Loc*, and *Random* defect patterns, so CNN-WDI was confused during the feature extraction of these defect patterns. Thus, the number of misclassified of *Local* defect patterns with *Donut, Edge-Loc*, and *Random* defect patterns was up to 2.1%, 1.1%, and 2.1%, respectively. Due to various reasons, some WMs contain multiple defect patterns on their surfaces. Therefore, the CNN-WDI model misclassified some of the defect classes with other defect patters. The *Edge-Loc* class was misclassified with *Local* and

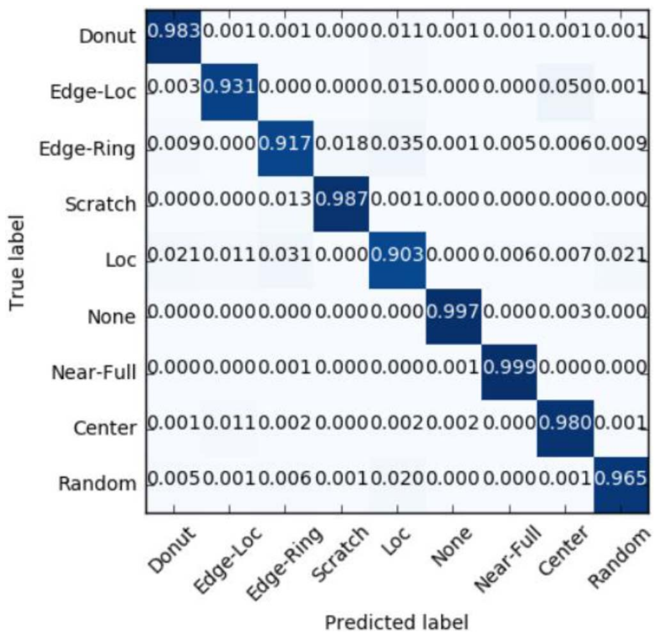

Fig. 4. A normalized confusion matrix of proposed CNN-WDI model.

Center classes with a ratio of 1.5% and 5.0%, respectively. The *Edge-Ring* class was misclassified with *Scratch and Local* classes with a ratio of 1.8% and 3.5%, respectively. The average classification result of the proposed model is up to 96.2%, which shows the importance of our model to classify WM defect patterns. The *None* defect class has no actual defect patterns in its wafer surface and this class contains the majority data like 85% of the original dataset of WM-811K. Thus, the higher classification result (i.e., 99.7%) of our model for the *None* defect class will boost the speed of testing process of WMs during semiconductor manufacturing.

Three more CNN classifiers (e.g., CNN-D, CNN-BN, and CNN-SD) with various hyperparameter settings were designed to evaluate the importance of different hyperparameters of the proposed model. The CNN-D model does not contain SD and BN methods, but only contains simple dropout method. The CNN-BN only contains BN but does not contain SD. The CNN-SD only contains spatial dropout but does not contain BN. The performances of all these CNN classifiers which contain various hyperparameter settings are compared with the

TABLE III
OVERALL PERFORMANCE COMPARISON OF VARIOUS CLASSIFIERS (%)

Classifier	Training Acc	Validation Acc	Testing Acc	Precision	Recall	F1-Score
CNN-WDI	98.9	**96.4**	**96.2**	**96.2**	**96.2**	**96.2**
CNN-D	97.6	95.5	95.2	95.2	95.2	95.2
CNN-BN	**99.4**	95.6	95.6	95.6	95.6	95.6
CNN-SD	98.6	94.7	94.8	94.8	94.8	94.8
VGG-16	82.3	80.0	80.1	80.3	80.1	79.9
ANN	95.9	95.9	72.0	95.2	95.9	95.4
SVM	91.3	91.0	32.6	87.5	91.0	88.0

Note: Boldface numbers denote the highest values of different performance measures and Acc accuracy

proposed CNN-WDI classifier which contains both BN and SD methods.

We also have compared the CNN-WDI classifier with state-of-the-art DL and ML classifiers like VGG-16, ANN, and SVM. The VGG-16 network is a well-known CNN model proposed to classify the image data with very high accuracy [32]. It composed of 16 layers including 13 Conv layer, 5 max-pooling layers, and 3 fully connected layers. Each Conv layer and first 2 fully connected layers have ReLU activation function, and the final layer has Softmax activation function for classification. The ANN model defines a relationship between input data and output value in the same way as derived from the animal brain [33]. We applied an ANN model with one input layer, one hidden layer with 100 neurons and ReLU activation function, and one output layer to classify the input data. The backpropagation algorithm and Adam optimizer were used to train and weight optimization of the ANN model. The SVM classifier is commonly used for image classification purpose especially for wafer defect classification [3]. The radial basis kernel function was implemented to handle the non-linear classification parameters due to complex image data. The penalty and stopping tolerance parameters were set to 1.0 and 0.001, respectively. The SVM is a binary classifier, so we applied one-vs-rest (OvR) method to classify multiple classes. We manually extracted Radon-based features by Radon transform from the raw wafer images to train ANN and SVM models. The same WM image dataset is used to train all the above classifiers as we used for CNN-WDI model.

Table III presents the performance comparison of all classifiers with overall performance measures. It shows that the CNN-WDI model outperformed all the other classifiers to classify the wafer defects and acquire the highest value of overall validation accuracy, testing accuracy, precision, recall, and F1-score of 96.4%, 96.2%, 96.4%, 96.2%, and 96.2%, respectively. Testing accuracy is the most important component of performance measures because it uses unseen data to evaluate a model. The proposed model improved the testing accuracy up to 20.1%, 33.6%, and 195.1% of VGG-16, ANN, and SVM, respectively. The CNN-WDI model also got the better results as compare to the CNN-D, CNN-BN, and CNN-SD, which proves the importance of hyperparameters used in proposed model. Although, ANN model showed very good performance measures and got better results of training accuracy, validation accuracy, and F1-score. However, we need manually extracted features to train the ANN and SVM model. Moreover, these classifiers performed very poorly for minority data sample defect classes such as *Local*, *Random*, *Scratch*, and *Near-Full* defect classes. The VGG-16 has a very high number of trainable parameters (i.e., 134.2 million of parameters) as compared to the proposed model which has only 2.7 million trainable parameters, still our model improved the training accuracy up to 18.6%. The training time of VGG-16 was up to 4.5×10^5, whereas CNN-WDI model took only 1.6×10^5 for training.

Table IV shows the testing accuracy performance comparison of CNN-WDI model with recently proposed wafer defect classification models such as WMFPR [3], DTE-WMFPR [5], and WMDPI [6], using the same dataset of WM-811K. As all defect patterns are equally important because each of these has a specific reason to occur. So, we cannot ignore any defect class during classification. It is clear from the table that deep learning-based CNN-WDI model outperformed all the previous models to classify the *Center*, *Donut*, *Edge-Loc*, *Local*, *Random*, *Scratch*, and *Near-Full* defect classes with the accuracy of 98.0%, 98.3%, 93.1%, 90.3%. 96.5%, 98.7%, and 99.9%, respectively. The classification accuracy of remaining defect classes is also very high such as above 92%. Whereas, all other classification models have poor accuracy to classify some of the defect classes. For example, WMFPR has 68.5% accuracy for *Local* class, WMDPI has 60.0% and 34.0% for *Local* and *Scratch* classes, respectively, and DTE-WMFRP has 83.5% for both *Edge-Loc* and *Local* classes. The average wafer defect classification accuracy of proposed model is 96.2%, which is improved up to 6.4% of the last highest average accuracy. Thus, these results show the importance of CNN-WDI model to classify wafer defect patterns.

Fig. 5 shows the comparison of training accuracy and loos with validation accuracy and loss of CNN-WDI model and VGG network. In Fig. 5(a), the training accuracy of CNN-WDI model increases suddenly in initial epoch and then gradually increases with increasing number of epochs. The behavior of validation accuracy is opposite because it decreased initially and then increases gradually and reaches up to 96.4%. But the training process of VGG-16 is very slow and its validation accuracy stopes increasing after achieving 80.0% as shown in Fig. 5(b). The same behavior can be observed for the training and validation losses of these models. The training and validation losses of proposed model decreased suddenly at initial stage and gradually decreased in next epochs and reached the value of 0.03 and 0.15, respectively, as shown in Fig. 5(c).

TABLE IV
COMPARISON OF DEFECT CLASSIFICATION RESULTS FOR DIFFERENT CLASSIFICATION MODELS (%)

Classification model	Center	Donut	Edge-Loc	Edge-Ring	Local	Random	Scratch	Near-Full	None	Average
CNN-WDI	**98.0**	**98.3**	**93.1**	91.7	**90.3**	**96.5**	**98.7**	**99.9**	99.7	**96.2**
WMFPR [3]	84.9	74.0	85.1	79.7	68.5	79.8	82.4	97.9	95.7	83.1
WMDPI [6]	86.0	77.0	77.0	**95.0**	60.0	94.0	34.0	97.0	**100.0**	80.0
DTE-WMFPR [5]	95.8	92.2	83.5	86.8	83.5	95.8	86.0	N/A	100.0	90.4

Note: Boldface numbers denote the highest values of testing accuracies for various classification models

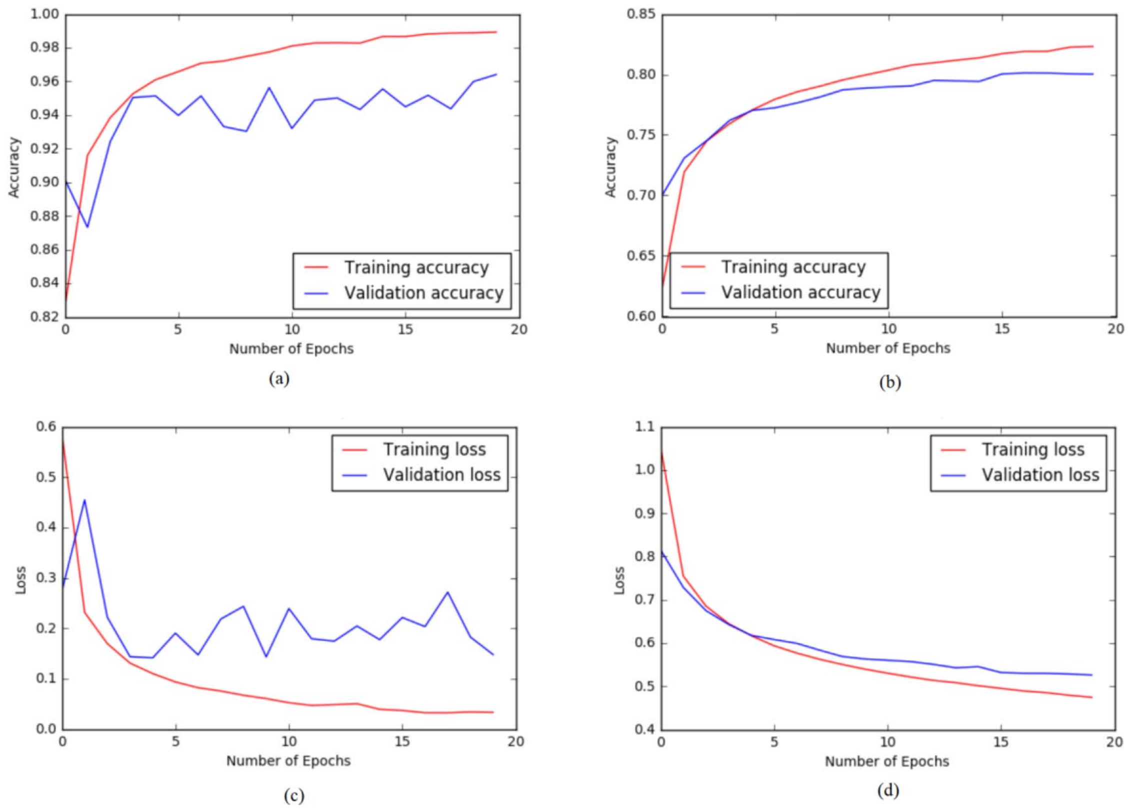

Fig. 5. The performance results of CNN-WDI model and VGG-16 model: (a) The accuracies of CNN-WDI, (b) The accuracies of VGG-16, (c) The losses of CNN-WDI, and (d) The losses of VGG-16.

Whereas, for the same number of epochs training and validation losses of VGG-16 network got the value of 0.48 and 0.80, respectively, as shown in Fig. 5(d).

V. CONCLUSION

In this paper, a deep learning-based CNN-WDI model was proposed to classify wafer map defects in the semiconductor fabrication process. Semiconductor engineers apply automatic wafer classification for early diagnosis of wafer defects without requiring specialized or empirical knowledge. Most of the previous wafer defect analysis had applied machine learning-based classification models, which required manual feature extraction and a lot of hyperparameter settings, whereas CNN model can automatically extract effective features of various defect classes. We have used a real wafer map dataset of WM-811K, that contains nine different defect patterns. This dataset is highly imbalanced, so we have applied data augmentation technique to solve this issue. The sequence of convolutional and pooling layers has applied to extract valuable features from raw wafer images. Batch normalization and spatial dropout methods have been used for regularization of the proposed model which improves the training speed and classification accuracy of the model. Our model has got very high performance for all defect classes as compared to previously proposed models such as WMFPR, DTE-WMFPR, and WMDPI using the same dataset, and achieved the classification accuracy of 96.2% on average. The CNN-WDI outperformed VGG-16, SVM, and ANN classifiers. For future work, we will extract multiple defects on the same wafer image to improve classification accuracy.

REFERENCES

[1] S. Cheon, H. Lee, C. O. Kim, and S. H. Lee, "Convolutional neural network for wafer surface defect classification and the detection of unknown defect class," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 2, pp. 163–170, May 2019, doi: 10.1109/TSM.2019.2902657.

[2] C. Wang, W. A. Y. Kuo, and H. Bensmail, "Detection and classification of defect patterns on semiconductor wafers," *IIE Trans.*, vol. 38, no. 12, pp. 1059–1068, Jan. 2006, doi: 10.1080/07408170600733236.

[3] M. Wu, J.-S. R. Jang, and J.-L. Chen, "Wafer map failure pattern recognition and similarity ranking for large-scale data sets," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 1, pp. 1–12, Feb. 2015, doi: 10.1109/TSM.2014.2364237.

[4] C. H. Jin, H. J. Na, M. Piao, G. Pok, and K. R. Ryu, "A novel DBSCAN-based defect pattern detection and classification framework for wafer bin map," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 3, pp. 286–292, Aug. 2019, doi: 10.1109/TSM.2019.2916835.

[5] 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 Trans. Semicond. Manuf.*, vol. 31, no. 2, pp. 250–257, May 2018, doi: 10.1109/TSM.2018.2806931.

[6] M. Saqlain, B. Jargalsaikhan, and J. Y. Lee, "A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 2, pp. 171–182, May 2019, doi: 10.1109/TSM.2019.2904306.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015, doi: 10.1016/j.neunet.2014.09.003.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, Jun. 2017, doi: 10.1145/3065386.

[9] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.*, vol. 30, no. 2, pp. 135–142, May 2017, doi: 10.1109/TSM.2017.2676245.

[10] T. Nakazawa and D. V. Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 2, pp. 309–314, May 2018, doi: 10.1109/TSM.2018.2795466.

[11] K. Kyeong and H. Kim, "Classification of mixed-type defect patterns in wafer bin maps using convolutional neural networks," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 3, pp. 395–402, Aug. 2018, doi: 10.1109/TSM.2018.2841416.

[12] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no, 1, pp. 1–21, Feb. 2015, doi: 10.1186/s40537-014-0007-7.

[13] Y. Tang, Y.-Q. Zhang, N. V. Chawla, and S. Krasser "SVMs modeling for highly imbalanced classification," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 1, pp. 281–288, Feb. 2009, doi: 10.1109/TSMCB.2008.2002909.

[14] Mirlab.org. (2018). *MIR Corpora*. Accessed: Jan. 4, 2019. [Online]. Available: http://mirlab.org/dataSet/public/

[15] S. H. Khan, M. Hayat, M. Bennamoun, F. A. Sohel, and R. Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3573–3587, Aug. 2018, doi: 10.1109/TNNLS.2017.2732482.

[16] G. Mariani, F. Scheidegger, R. Istrate, C. Bekas, and C. Malossi, "BAGAN: Data augmentation with balancing GAN," Jun. 2018. [Online]. Available: https://arxiv.org/abs/1803.09655.

[17] E. D. Cubuk, B. Zoph, D. Man, V. Vasudevan, and Q. V Le, "AutoAugment: Learning augmentation strategies from data," Apr. 2019. [Online]. Available: https://arxiv.org/abs/1805.09501.

[18] P. Kim, *MATLAB Deep Learning With Machine Learning, Neural Networks and Artificial Intelligence*. Heidelberg, Germany: Springer, 2017, doi: 10.1007/978-1-4842-2845-6.

[19] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," *Gen. Program. Evolvable Mach.*, vol. 19, no. 1, pp. 305–307, Jun. 2018, doi: 10.1007/s10710-017-9314-z.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[21] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Int. Conf. Mach. Learn. (ICLM)*, Jun. 2010, pp. 807–814.

[22] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *J. Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, Jun. 2017, doi: 10.1162/NECO_a_00990.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Mar. 2015. [Online]. Available: https://arxiv.org/abs/1502.03167.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jun. 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[25] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," Jul. 2012. [Online]. Available: https://arxiv.org/abs/1207.0580.

[26] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler, "Efficient object localization using convolutional networks," Jun. 2015. [Online]. Available: https://arxiv.org/abs/1411.4280.

[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," Jan. 2017. [Online]. Available: https://arxiv.org/abs/1412.6980.

[28] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," Feb. 2017. [Online]. Available: https://arxiv.org/abs/1609.04836.

[29] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," Mar. 2015. [Online]. Available: https://arxiv.org/abs/1603.04467.

[30] Keras.io. (2018). *Keras Documentation*. Accessed: May 21, 2019. [Online]. Available: https://keras.io/

[31] Jupyter.org. (2019). *Project Jupyter*. Accessed: May 15, 2019. [Online]. Available: http://jupyter.org/

[32] K. Simonyan and Z. Andrew, "Very deep convolutional networks for large-scale image recognition," Apr. 2015. [Online]. Available: https://arxiv.org/abs/1409.1556.

[33] R.-S. Guh, "On-line identification and quantification of mean shifts in bivariate processes using a neural network-based approach," *Qual. Rel. Eng.*, vol. 23, no. 3, pp. 367–385, Apr. 2007, doi: 10.1002/qre.796.