Convolutional Neural Network for Wafer Surface Defect Classification and the Detection of Unknown Defect Class

Sejune Cheon, Hankang Lee, Chang Ouk Kim, and Seok Hyung Lee

***Abstract*—An automatic defect classification (ADC) system identifies and classifies wafer surface defects using scanning electron microscope images. By classifying defects, manufacturers can determine whether the wafer can be repaired and proceed to the next fabrication step. Current ADC systems have high defect detection performance. However, the classification power is poor. In most work sites, defect classification is performed manually using the naked eye, which is unreliable. This study proposes an ADC method based on deep learning that automatically classifies various types of wafer surface damage. In contrast to conventional ADC methods, which apply a series of image recognition and machine learning techniques to find features for defect classification, the proposed model adopts a single convolutional neural network (CNN) model that can extract effective features for defect classification without using additional feature extraction algorithms. Moreover, the proposed method can identify defect classes not seen during training by comparing the CNN features of the unseen classes with those of the trained classes. Experiments with real datasets verified that the proposed ADC method achieves high defect classification performance.**

***Index Terms*—Automatic wafer surface defect classification, Deep learning, Convolutional neural network, Unknown defect detection, *k*-nearest neighbors algorithm.**

I. Introduction

DURING semiconductor fabrication, integrated circuits (ICs) are made by creating circuit structures on many layers of a single wafer and interconnecting the structures using wires. The important process steps needed to complete each circuit layer are photolithography, etching, deposition, ion implantation, diffusion, and chemical mechanical planarization. To produce a high-density IC, the wafer surface must be extremely clean, and the circuit layers fabricated on the previous wafer should be aligned. The high-density structure may collapse if these conditions are not satisfied.

One important method of preventing collapse is to take scanning electron microscope (SEM) images of the wafer surface after the completion of each circuit layer (particularly between the etching and deposition steps) and inspect whether there are any particles (e.g., rock and ring shapes), flaws (e.g., spots and scratches), or irregular connections caused by misaligned electronic circuits stacked on layers (see Fig. 1). By classifying wafer surface defects, manufacturers can determine the defect-introducing process steps and whether the wafer can be repaired and proceed to the next fabrication step. Repairable defects are limited to particle-type defects which can be reworked by cleaning the surface with an air blower. In addition to particles, defective dies are extracted from the wafer and disposed of.

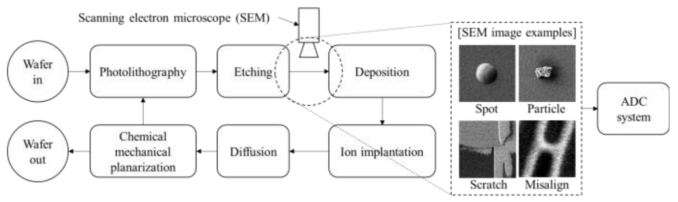

Fig. 1. Main IC fabrication steps and ADC system.

As an effort to improve product quality while reducing manufacturing and labor costs, semiconductor manufacturers have introduced automatic defect classification (ADC) systems. An ADC system scans the wafer surface, collects the coordinates of suspicious areas where defects may exist, places its camera lens at the center of each coordinate and takes an SEM image patch. Current ADC systems can identify defects automatically (defect detection), and performance is high. It is very unlikely that the SEM image will contain multiple defects because the SEM sensing field is microscopic. However, defect classification accuracy is poor and a manual inspection process is also needed for classification. For this reason, research is still needed to achieve high-performance ADC systems with improved product yield and reduced labor cost of inspection personnel.

The detection and classification of microscopic defects on semiconductor wafers have not been adequately addressed in the related literature. The study of defect control concentrates on defect inspection and detection [1]. Furthermore, current ADC approaches apply a series of image recognition and machine learning techniques to find features for defect classification. Chou *et al.* [2] developed a system that applies a decision tree and neural network to classify defects on wafer-

This work was supported in part by the NRF through the Ministry of Science, ICT and Future Planning, South Korea under Grant NRF-2016R1A2B4008337 and the Graduate School of YONSEI University Research Scholarship Grants in 2018.

S. Cheon, H. Lee, and C. O. Kim are with the Department of Industrial Engineering, Yonsei University, Seoul, Republic of Korea (e-mail: kimco@yonsei.ac.kr).

S. H. Lee is with SK Hynix, Icheon, Republic of Korea.

level chip-scale package images. As a preprocessing step, the system measures several features of wafer surface images, such as the size, shape, location and color of possible defects. Then, the system inputs the features into classifiers. Chang *et al.* [3] used a hybrid inspection approach. They initially divided defect images into two types—high-intensity and low-intensity defect images—using morphological operations and a simple threshold method. For high-intensity defect images, researchers applied a support vector machine (SVM) with extracted input features such as smoothness and complexity of texture. For low-intensity defect images, researchers developed a simple boundary defect-detection algorithm that uses a contour of the image and the Hough transform. Su *et al.* [4] proposed a neural network approach for semiconductor wafer post-sawing inspection. In this approach, wafer images should be downsized using a mask to reduce the input-node size of the neural network. Three neural network models were tested to determine whether the wafer is normal or faulty.

Recently, deep learning has gained growing attention because it is able to automatically extract compact features from complex and high-dimensional data (e.g., images and sensor signals). Lee *et al*. [5] proposed a stacked denoising autoencoder model for the fault detection of a worksite photolithography tool. They processed sensor signals using the model to extract fault detection features that are robust against sensor measurement noise. In particular, the convolutional neural network (CNN) model was widely adopted because it showed excellent performance for classifying image data. Industries have introduced CNN for crack inspection of civil infrastructures [6], surface defect classification of steel sheets [7], and defect recognition of spring-wire sockets [8]. The semiconductor industry has also been interested in CNN applications for process improvement. Lee *et al.* [9] modified the classical CNN structure to find multivariate process faults using sensor signal data and to diagnose the source of the faults. Lee *et al.* [10] proposed a hybrid CNN model, incorporated with a recurrent neural network, for a virtual metrology that is robust to process-drift in chemical mechanical planarization processes. Nakazawa and Kulkarni [11] applied CNN for wafer map classification. However, CNN models for wafer surface defect classification have not been addressed.

In this paper, we propose a CNN-based ADC method. The proposed method has two contributions. First, the method is specifically designed so that the CNN architecture achieves high classification performance for known defect classes. Second, the method is able to classify unknown defects as an ‘Unknown’ class without retraining the CNN by using a *k*-nearest neighbors (*k*-NN) algorithm that is implemented in the feature space created by the CNN training result.

CNN is a supervised model, and thus, it is not possible to detect classes that have not been seen during training. Thus, CNN is limited for ADC because the wafer defect class set is not closed, i.e., defect classes can be observed as process recipes and equipment are changed. Existing CNN studies focus on the classification of predetermined classes. As such, a trained model will incorrectly label an unseen defect as one of classes seen during training. To make the model correctly classify new defects, retraining with all data, including the new data collected, is necessary, which is time consuming and inefficient because CNN retraining requires sufficient data, and thus, it is not possible to detect a new class until a sufficient amount of data for the class are accumulated.

We tested the performance of the proposed ADC method with real wafer surface defect datasets obtained from a semiconductor manufacturing field site and demonstrated that the CNN for the ADC method outperformed multilayer perceptron (MP), SVM and stacked autoencoder (SAE) [12], which is another widely used neural network model with a deep structure. The classification accuracy of the CNN with five different defect classes was 96.2%. In addition, we tested 30 defects unknown during CNN training to verify whether the unknown defects can be classified correctly as ‘Unknown’. The result showed that only two images were misclassified, and 28 images were declared as the ‘Unknown’ class.

The remainder of this paper is organized as follows. Section 2 introduces the principles of MP, CNN, and SAE. Section 3 presents the proposed ADC method and section 4 evaluates its performance with field data and analyzes the results. Section 5 draws conclusions and discusses topics for future research.

II. Deep Learning Models

Understanding the operating principle of deep learning requires basic knowledge of MP because CNN is an extension of MP and operates based on the same principle. This section introduces the operating principle of MP, illustrates the structure of CNN, and briefly introduces SAE, a performance comparison model in the experiment.

A. MP

An MP has a single input layer, multiple hidden layers, and a single output layer. Each layer has multiple nodes, and the nodes of each layer are fully connected to those of its neighboring layer. The connections between layers have weights that must be determined. The input layer receives input data for classification; the hidden layers find classification features based on the input data and weights between the input and hidden layers; the output layer produces the classification result based on the features and weights between the hidden and output layers. The training of the weights between layers corresponds to the determination of the features effective for the classification of input data. Given the training data, a backpropagation algorithm [13] seeks the optimal weights that maximize classification performance.

B. CNN

CNN is a variant of MP, and its structure is specifically designed to learn the classification features of images [14]. Fig. 2 shows a basic CNN architecture. It has two types of hidden layers: convolutional layers and pooling layers. Convolutional layers identify primitive geometric patterns, such as local edges, endpoints, and corners, in input images, and pooling layers find location-invariant features of large patterns that combine the primitive patterns of convolutional layers. The two hidden layers are responsible for feature learning. CNN has a

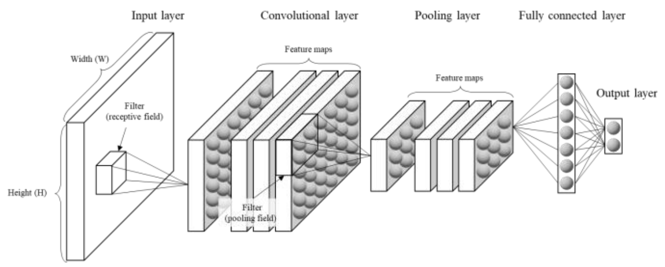

Fig. 2. Basic structure of the convolutional neural network.

fully connected layer to classify the input image based on the features found by the two hidden layers. Because images generally contain many local and distorted patterns, CNNs form a deep learning architecture [15], in which convolutional and pooling layers are stacked to find global features that are effective for classification.

As shown in Fig. 2, when a gray-level wafer image with a size of $W \times H$ is fed into the input layer, the CNN connects the local regions of the input image defined by the filter (also referred to as the receptive field) with nodes in the convolutional layer. Specifically, a node in the convolutional layer is connected to all input pixels within a filter with a size of $F \times F$, and the connections have F^2 weights and one bias term. Such a connected structure allows the node to detect a pattern of the input pixels within the filter. Starting from the top left, the filter goes through convolution with the entire input image by sliding to the right (or down) with a fixed stride length S. The series of convolution operations results in the creation of a two-dimensional node array that connects to the input image. This array is called a feature map. The width and height of the map are $(W-F)/S+1$ and $(H-F)/S+1$, respectively. The convolutional operation is expressed by (1) [14].

$$y_{ij} = \sigma\left(\sum_{r=1}^{F}\sum_{c=1}^{F} w_{rc}x_{(r+i\times S)(c+j\times S)} + b\right),\quad 0 \le i \le \frac{H-F}{S}, 0 \le j \le \frac{W-F}{S}. \tag{1}$$

In (1), y_{ij} indicates the output value of node (i,j) in the feature map; $x_{(r+i\times S)(c+j\times S)}$ is the input data element with the coordinate $(r+i\times S, c+j\times S)$; w_{rc} and b are the weight of the coordinate of (r,c) on the receptive field and the bias, respectively; and σ denotes any nonlinear activation function used to extract features of the input data. Representative activation functions include the rectified linear unit (ReLU), the hyperbolic tangent, and the sigmoid function.

According to (1), all nodes in the feature map have identical weights w_{rc} $(r,c=1,\dots,F)$ and bias term b. Weight sharing means that the CNN detects the same type of feature in different image areas [14]. A convolutional layer may have a different number of feature maps that depend on image complexity for classification. Each feature map has its own weight array and bias term. Therefore, if the input pixels within a certain filter are connected to K feature maps, the CNN will find K different features within the same local region.

The pooling layer is usually located behind the convolutional layer. The pooling layer has the same number of feature maps as the preceding convolutional layer, and a feature map in the layer is connected to a feature map in the convolutional layer; however, the size of the feature map in the pooling layer is smaller than that in the convolutional layer. In the pooling layer, each node of a feature map summarizes the information in the $P \times P$ node array (also referred to as a pooling filter) of the connected feature map in the convolutional layer. Max-pooling is a popular summarization procedure. It outputs the maximum value of the nodes within the filter of a convolutional feature map. Thus, the pooling layer downsamples the convolutional feature map. There is no actual learning process in this layer. Average pooling and L^2 norm pooling are also widely used.

Pooling enables the extraction of location-invariant and translation-invariant features [14]. Therefore, the learned feature is activated by the activation function even under small translations of the input image. Another advantage of pooling is that it can reduce the learning parameters (i.e., weights and bias terms) required for the next layer. Typically, a pooling layer is inserted periodically between successive convolutional layers in a CNN architecture. Such a layer architecture will combine local features found from shallow layers at deeper layers, which will enable the discovery of complex global features that are useful for classifying defect images. The layer depth can affect the classification performance at the output layer. Thus, the depth should be determined based on the complexity of the target image to analyze.

The final layers of the CNN are the fully connected and output layers. The purpose of the full connection is to combine the global invariant features received from previous layers and use them as information for classification. The output layer is composed of as many nodes as there are class labels and outputs the class score for the input image using the softmax function in (2).

$$P(Y=i|\boldsymbol{x},\boldsymbol{W}_i,b_i) = \frac{e^{\boldsymbol{W}_i^T x+b_i}}{\sum_j e^{\boldsymbol{W}_j^T x+b_j}}. \tag{2}$$

In (2), $\boldsymbol{x}$ refers to the input vector (i.e., wafer image), and $\boldsymbol{W}_i$ and b_i refer to the weight vector and bias term connected to output node i, respectively. Using the equation, each node of the output layer calculates the probability that the input vector $\boldsymbol{x}$ belongs to class i. The CNN selects the class with the highest probability as the classification result.

Similar to MP, CNN uses the backpropagation algorithm, which learns the weights and bias terms to minimize a loss function. This study uses the categorical cross entropy (CCE) as the loss function [14] for minimization.

$$CCE = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} 1_{y_i\in c}\log p_{model}[y_i \in c]. \tag{3}$$

In (3), N and C are the numbers of training data and class labels, respectively, and $1_{y_i\in c}$ is the indicator function, which is one if the actual class y_i of data i matches class c and zero otherwise; $p_{model}[y_i \in c]$ is the prediction probability by the CNN model.

C. SAE

SAE is constructed based on autoencoder (AE). AE is an MP trained to reproduce input data at the output layer [14]. The network is composed of a single input layer, a hidden layer, and an output layer. The input and output layers are the same size. AE has two parts: an encoder and a decoder. The encoder, which is the network between the input and hidden layers, encodes the input data in a reduced dimension of the hidden layer (the number of hidden nodes is usually less than that of input nodes). The decoder, which corresponds to the hidden and output layers, reconstructs the input data at the output layer using the encoded information (features) of the hidden layer. AE training determines the weights of the hidden layer such that the disparity between the input and the output of the network is minimized. Thus, the main purpose of AE is to represent input data in the reduced feature space.

An SAE stacks AEs in layers to represent complex input data in a feature dimension that is smaller than the original input dimension. It employs a layer-by-layer pretraining method. For an SAE with two hidden layers, for example, the first AE is trained using the backpropagation algorithm. Then, the output layer of the trained first AE is removed, making the hidden layer of the first AE become the input layer of the second AE. The weights of the second AE are updated via backpropagation. During SAE training, the features (i.e., outputs of hidden nodes) in each hidden layer are fed into the next hidden layer, thereby creating more highly abstracted features. Finally, fine-tuning the entire network is followed by the training method to achieve higher classification performance.

With respect to network architecture, an SAE is the same as an MP with multiple hidden layers. SAE has an input layer, hidden layers, and an output layer. The difference between the two lies in the weight training mechanism. MP employs entire-network training, i.e., the weights of all layers are trained using the backpropagation algorithm, while SAE adopts layerwise training to resolve the vanishing gradient problem of network training [5], i.e., only the weights of a few hidden layers close to the output layers are trained and those far from the output layers are not properly trained.

CNN and SAE are both deep learning models. Hidden layers can be stacked deep to achieve high performance. However, CNN is specifically designed for image classification by adopting the weight sharing mechanism. A filter defining a local area of the input image has the same weights; as a result, all hidden nodes of a feature map share the same weights, thereby enabling the same local feature to be found over the entire input image. Different filters produce different feature maps and thus find different local features. In contrast, SAE and MP do not adopt weight sharing. All connections between input nodes and hidden nodes have unique weights. Each hidden node is a feature extractor that finds a feature over the entire input image.

III. Proposed Method

A. CNN Architecture

CNN has various network architectures depending on the fields in which it is applied. The depth of the stacked layers can be adjusted according to the complexity of the image to be analyzed. For ADC, we propose a CNN architecture that consists of one input layer, four convolutional layers and two pooling layers (stacking a pair of convolutional–convolutional–pooling layers), one fully connected layer, and one output layer. The layout of the CNN architecture is depicted in Fig. 3.

The number of feature maps increases with increased depth of the convolutional layer. However, the size of the feature map reduces due to the pooling layer if the network layer is deep. The proposed architecture adopts 32 feature maps to the first stacking pair (convolutional–convolutional–pooling layers) and 64 feature maps to the second stacking pair. This architecture increases the ability of the CNN to capture various geometric defect features with a rich set of feature maps and avoid information loss caused by reduced feature map size. The filter sizes of the convolutional and pooling layers are designed small to make the CNN find detailed features that are useful for classification in defect images. The ReLU function, which is the default recommendation in modern deep neural networks, is applied as the activation function [14]. The function is applied to all nodes of the CNN except the output layer. In addition, the dropout technique [16], which is a regularization method, is applied to prevent possible overfitting to the training data during the learning process. The details of the proposed configuration are listed in Table I.

B. Detection of Unknown Defects

CNN trained with image data of predetermined defect classes cannot guarantee long-lasting, high classification performance in dynamic manufacturing environments in which new defects are irregularly generated. In this study, we propose a *k*-NN algorithm that can detect unknown defects using a trained CNN. As explained previously, CNN is composed of feature-learning and classification steps. The last layer of feature learning is the fully connected layer, and the size of the layer is much less than that of the input image. Thus, the fully connected layer produces a feature vector representing the original input image

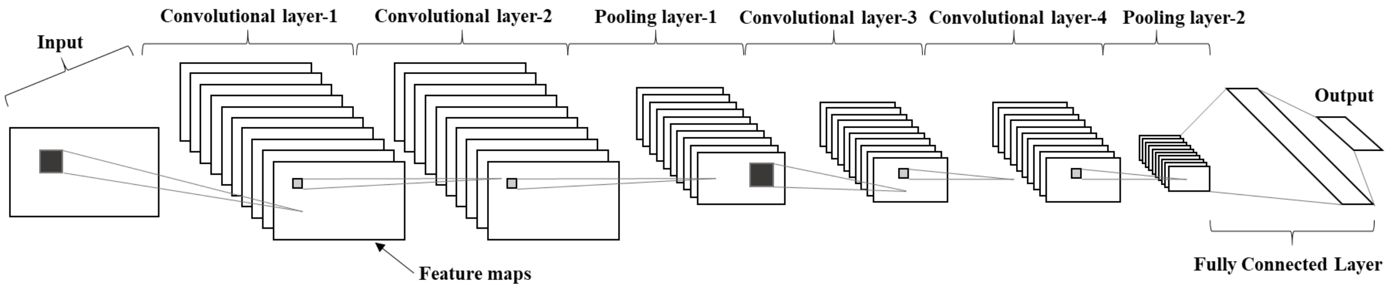

Fig. 3. CNN architecture for the ADC method.

TABLE I
PROPOSED CNN CONFIGURATION

CNN parameter	Value
Number of convolution layers	4
Filter size	3 x 3
Number of pooling layers	2 max-pooling
Pooling filter size	2 x 2
Number of feature maps	32/32/64/64
Number of fully connected layers	1
Number of nodes in the fully connected layer	512
Activation function	ReLU
Regularization method	Dropout
Classification function of the output layer	Softmax
Loss function	Categorical cross entropy

in the reduced feature space. If the feature vectors of all training images are plotted in the space, the images of the same defect class will form a cluster because the trained CNN outputs similar feature vectors for defects of the same class to maximize classification performance (a three-dimensional (3D) clustering plot is shown in Fig. 6 of the experiment section). The *k*-NN algorithm detects anomalies in the feature space. When an unknown defect image is input to the trained CNN, the CNN generates its feature vector and *k*-NN conducts a membership test for each cluster. If the image is determined not to belong to any cluster, it is declared part of the 'Unknown' class. The detailed algorithm is presented next.

1) *Clustering and threshold setting for membership testing*

Step 1. Filter out the defect images in the training dataset that are misclassified by the trained CNN. In addition, replace the activation function of the fully connected layer from ReLU with a sigmoid function so that the feature vector has component values between 0 to 1 (scaling effect). For correctly classified training images, generate feature vectors using modified CNN.

Step 2. Construct a *k*-NN classifier for each cluster (e.g., if there are five classes, construct five classifiers). In detail, for each image of a specific class, calculate the total squared distance between the feature vector of the image and its k nearest neighbors in the same class.

$$D_i^2 = \sum_{j=1}^{k} d_{ij}^2. \tag{4}$$

In (4), d_{ij}^2 is the Euclidian distance between the i^{th} image and the j^{th} nearest neighbor, and D_i^2 is the total squared distance of the i^{th} image. From each cluster, 100(1-α)% confidence limit (α denotes significance level) of the total squared distance distribution is set to the threshold for the membership test. The distance distributions of predetermined classes may overlap because some defect classes look similar. Thus, the threshold in step 2 allows the classification error rate, α. If the images in each cluster are distributed according to a normal distribution in the feature vector space, more rigorous threshold setting using the Mahalanobis distance that considers the density of data is possible. However, the normality assumption is not reasonable in this study.

2) *Membership test*

Step 1. For a test defect image, modified CNN extracts its feature vector.

Step 2. For each cluster, calculate the total squared distance D_{test}^2 and compare D_{test}^2 with the threshold of the cluster. If D_{test}^2 exceeds the thresholds of all clusters, the membership test fails, and the test image is declared as 'Unknown'. Otherwise, the test image is input into the trained CNN (not the modified one) to receive its label from among the predetermined classes. The *k*-NN detects anomalies (data that do not belong to predetermined classes). When a test image is not an anomaly, the CNN has a superior classification power than *k*-NN. Thus, step 2 uses CNN for defect classification.

IV. EXPERIMENT

A. *Datasets*

Wafer surface defect images were sampled from an actual inspection facility in the manufacturing field to evaluate the performance of the proposed ADC method. Each image was 160×160. We prepared two defect image datasets: Dataset-TT (train & test) and Dataset-UN (unknown). Dataset-TT was for training and testing the proposed CNN, and Dataset-UN was for evaluating the detection ability of the unknown defect images. Dataset-TT has 2,123 images of five defect classes (spot: 687; rock-shaped particle: 235; ring-shaped particle: 912; misalignment: 137; scratch: 152). Sample images of these defects are shown in Fig. 4. Dataset-UN has 30 defect images of classes not used for the CNN training.

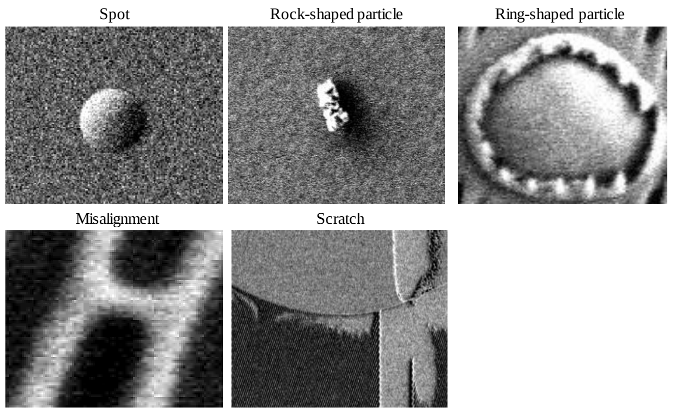

Fig. 4. Sample images of wafer surface defects (Dataset-TT).

B. *CNN Training*

Dataset-TT was divided into training, validation, and test subsets at proportions of 70%, 15%, and 15%, respectively. The training data were used to learn the CNN parameters that minimized the loss function. However, the training data size was not sufficient for training the CNN without overfitting because a large number of parameters (weights and bias terms) had to be determined. Thus, we tripled the amount of training image data by rotating each input image a random degree in the range [0, 180] and flipping the input image along the horizontal and vertical axes. The mini-batch stochastic gradient descent [17] method (learning rate: 0.001; batch size: 32) was used for CNN parameter learning.

Validation data were used to evaluate the classification accuracy of the trained CNN after each update of the CNN parameters using the augmented training data. To create a classifier with robust performance, the CNN should not exhibit the best performance for the training data but for the validation

TABLE II
CLASSIFICATION PERFORMANCE OF THE CNN FOR DATASET-TT

Defect class	Training dataset (70%)				Validation dataset (15%)				Test dataset (15%)			
	Total	Correct	Incorrect	Accuracy	Total	Correct	Incorrect	Accuracy	Total	Correct	Incorrect	Accuracy
Spot	480	477	3	99.4%	101	99	2	98.0%	106	104	2	98.1%
Rock-shaped particle	154	154	0	100%	38	38	0	100%	43	39	4	90.7%
Ring-shaped particle	643	638	5	99.2%	135	134	1	99.3%	134	128	6	95.5%
Misalignment	105	105	0	100%	18	17	1	94.4%	14	14	0	100%
Scratch	104	103	1	99.0%	26	26	0	100%	22	22	0	100%
Overall	1486	1477	9	99.4%	318	314	4	98.7%	319	307	12	96.2%

data. Finally, the test data were used to conduct the final performance investigation after the training process was complete. The experiment was conducted with a workstation computer with the following hardware specifications: CPU Intel(R) Xeon(R) E3-1231 v3 @ 3.40 GHz, 24 GB RAM, and NVIDIA GeForce GTX 1060 6GB.

C. CNN Classification Results and Analysis

The classification results are provided in Table II. The average classification accuracy of the trained CNN was 96.2% for the test data of Dataset-TT. The CNN misclassified only 12 of 319 test defect images. The accuracy was above 90% for all five defect classes. In addition, for the misalignment and scratch classes, CNN performance was perfect. The poorest classification accuracy was 90.7% for the rock-shaped particle class. However, the CNN misclassified only four images.

We conducted a comparison experiment with three different classifiers: SAE, MP, SVM with the radial basis kernel function (SVM-rbf). The classifiers accept full-size images as inputs, and each pixel of the 160×160 defect image corresponded to an input variable, resulting in 25,600 input dimensions. In addition, the performances of MP and SVM-rbf were tested using four preprocessed input features by the edge detection algorithm in Kuo *et al*. [18], instead of using the full-size image. The comparison results are provided in Table III, in which the proposed CNN exhibited better performance than the five comparison classifiers.

TABLE III
RESULTS OF THE COMPARISON EXPERIMENT

Classifier	Train accuracy	Valid accuracy	Test accuracy	CPU times in training	CPU times in testing
CNN	99.4%	98.7%	96.2%	42856	1.813
SAE	99.1%	94.0%	91.8%	49471	0.781
MP	99.9%	93.4%	92.8%	22757	1.313
SVM-rbf	100%	94.3%	92.5%	26485	125.516
MP (extracted feature)	53.1%	56.3%	55.2%	3970	0.016
SVM-rbf (extracted feature)	66.8%	62.9%	62.4%	145	0.203

SAE has one input layer, two encoding hidden layers with 2,560 and 1,280 nodes, respectively, one decoding hidden layer with 2,560 nodes, and one output layer. The network was designed to learn compressed feature vector whose size was 20 times smaller than the input dimension size. For the test data of Dataset-TT, the trained SAE achieved 91.8% classification accuracy. The result was inferior to the accuracy of the CNN by 4.4%.

The MP tested in this study has one input layer, two hidden layers consisting of 6,400 and 3,200 nodes, respectively, and one output layer. The number of nodes in each hidden layer were selected to extract as rich a set of classification features as possible. To do so, the size of the first hidden layer was set to a quarter of input data size, and the size of the second hidden layer was set to a half of the first hidden layer size. The accuracy for the test data was 92.8% which was similar to that of SAE but still lower than the CNN accuracy (by 3.4%).

From the network design perspective, the number of nodes in SAE and MP was selected via trial and error to attain the best accuracy.

SVM is a popular machine learning model that shows robust performance in most application areas including product surface defect classification. SVM is basically a binary classifier; thus, the one-versus-rest rule was applied for multiclass classification [19]. The radial basis kernel function, which creates nonlinear classification boundaries, was adopted because the input images were highly complex and because nonlinear boundaries were necessary for classifying defects. The kernel function required two parameters: cost and gamma. A sequential parameter search was conducted, which found the best values: 10 for the cost parameter and 0.001 for the gamma parameter. In Table III, the performance of SVM-rbf was comparable to those of SAE and MP; however, it performed worse than CNN.

The performance result of applying the feature extraction method prior to MP and SVM-rbf training was poor. We can conclude from the result that the feature extraction method failed to find representative defect features of the images in Dataset-TT. Overall, the results of the experiment support the claim that the proposed CNN model is superior to existing models for wafer surface defect classification.

The last two columns of Table III showed CPU times in seconds for training and testing the machine learning models. For training, MP and SVM-rbf using preprocessed features as training inputs took approximately eight times less than using the original defect images. CNN and SAE required long training times to find effective features hidden in the images. However, the test times of the trained CNN and SAE were very short (less than two seconds for the classification of a defect image).

The saliency maps of five different defect images are shown in Fig. 5. The maps illustrate which portions of the images the trained CNN considered critical for classifying defects. The saliency map computes the gradient values of the output class

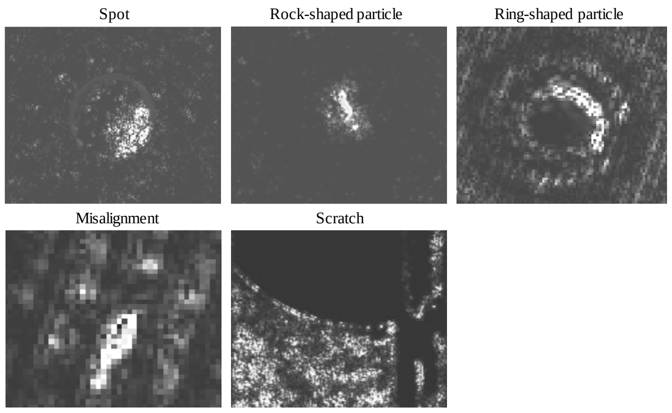

Fig. 5. Saliency maps of five sample defect images.

score with respect to input image pixel intensity. For a given test image of a specific class, the image is input into a trained CNN, and the corresponding output class score is obtained at the classification layer. Then, backpropagation is performed once to obtain the gradient of the output class score for each input pixel. The saliency map visualizes the gradient values of all input pixels. The higher the gradient value is, the more the pixel is activated for classification. The detailed logic of the saliency map is presented in Simonyan *et al.* [20].

The bright pixels of the saliency map are the locations on which the trained CNN focuses. Fig. 5 illustrates the locations corresponding to the wafer areas where the defects were positioned. This result means that the CNN was capable of successfully locating the positions of defect occurrences and shows which parts of the pixels the CNN considered features for classifying defects.

It was observed from the saliency maps and the classification performance of the trained CNN that CNN can capture high-quality classification features. To see how the classification features are formed, we visualized feature vectors extracted from the fully connected layer of the CNN. Because of the dimension of feature vector 512, AE was used to project the vector to 3D space for visualization. The AE consists of an input layer with 512 nodes, a hidden layer with three nodes, and an output layer with 512 nodes. The AE was trained with CNN feature vectors of images in Dataset-TT. The projected feature vectors (outputs of the hidden nodes) are plotted in Fig. 6, which shows that the feature vectors of each defect class formed a cluster; therefore, the CNN successfully found information in defect images to achieve the goal of defect classification.

D. *k-NN Test Result for Unknown Defects*

The detection performance of new defect images was tested using Dataset-UN. As shown in Fig. 6, clusters are separated well, which suggests that it is sufficient to use one nearest neighbor to classify a new defect image. Based on the observation, we set k to one. Note that the proposed *k*-NN is different from the classical nearest neighbor method, which assigns the label of the nearest class to the new image; the proposed *k*-NN does not assign the label of the nearest class to the new image if the image is far from that class.

For each cluster (class), the squared distances between

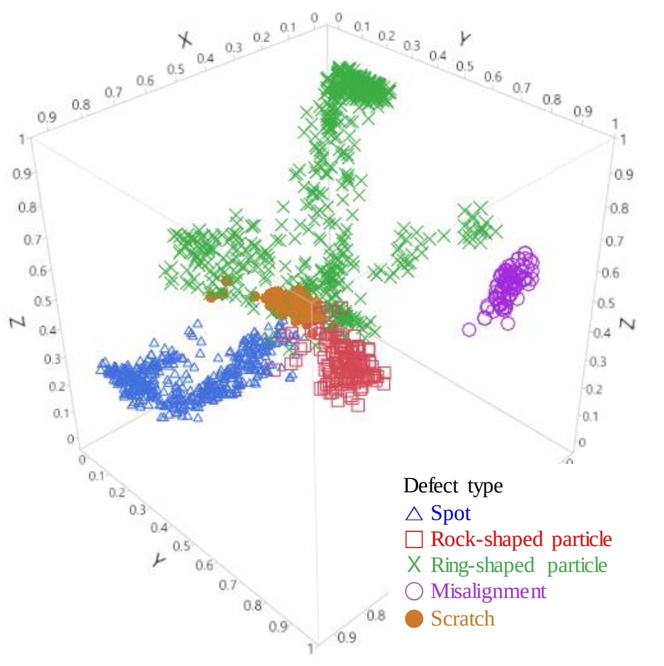

Fig. 6. Image feature vectors of Dataset-TT in 3D autoencoder output space.

member images and the nearest neighbors were calculated in the feature vector space, and the 90% percentile (α=0.1) of the empirical distribution of the total squared distance was set to the threshold for membership testing. We calculated five thresholds because Dataset-TT has five different defect classes. Table IV shows the thresholds and test results.

TABLE IV
k-NN TEST RESULTS FOR THE DATASET-UN

Defect class	Threshold value	Number of images that exceeds threshold
Spot	6.133	30
Rock-shaped particle	34.083	30
Ring-shaped particle	23.812	28
Misalignment	1.167	30
Scratch	43.700	30

Only 2 of 30 defect images were not assigned the ‘Unknown’ label. The error was because, for misclassified test images, the trained CNN outputted comparable feature vectors to that of the ring-shaped particle. As shown in Fig. 7, the shapes of the three images look similar.

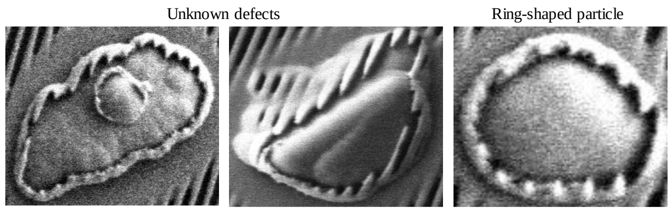

Fig. 7. Misclassified defect images vs. ring-shaped particle image.

In summary, the experiments demonstrated that the CNN was capable of extracting effective defect features from wafer surface images without any preprocessing steps. The features can be used to not only classify defects of classes known from training but also detect unknown class images if the proposed *k*-NN test is combined with the CNN.

V. Conclusion

This study proposed a new ADC method that combined CNN and *k*-NN. CNN can extract effective features for wafer surface defect classification without using other feature extraction algorithms. The performance evaluation experiment conducted using real defect image datasets yielded a classification accuracy of 96.2% on average. CNN outperformed SAE, MP and SVM-rbf.

Due to the dynamic nature of semiconductor manufacturing environments, wafer surface defect classes cannot be predetermined. Existing ADC methods misclassify new defects as one of the predetermined classes. In contrast, the proposed *k*-NN algorithm clusters CNN feature vectors and conducts membership testing for a new defect based on the total squared distance between the image and its *k*-nearest neighbors in the feature vector space. From the experiment, we identified that clustering was effective for detecting new defects.

Although the ADC method is appropriate for detecting new defects, the field application needs CNN retraining to ensure high performance over time. Incremental training is a solution. This strategy applies CNN and *k*-NN for a time window and, after that, CNN is retrained with the entire historical dataset. This approach ensures steady ADC performance but demands a heavy computation burden because defect data are continually accumulated. In the future, ADC should be based on an unsupervised clustering model that creates a cluster whenever a new class image is collected.

References

[1] S. H. Huang and Y. C. Pan, “Automated visual inspection in the semiconductor industry: A survey,” *Comput. Ind.*, vol. 66, pp. 1–10, 2015.

[2] P. B. Chou, a. R. Rao, M. C. Sturzenbecker, F. Y. Wu, and V. H. Brecher, “Automatic defect classification for semiconductor manufacturing,” *Mach. Vis. Appl.*, vol. 9, no. 4, pp. 201–214, 1997.

[3] C. Chang, J. Wu, and Y. Wang, “A hybrid defect detection method for wafer level chip scale package images,” *Int. J. Comput. Consum. Control*, vol. 2, no. 2, pp. 25–36, 2013.

[4] C. T. Su, T. Yang, and C. M. Ke, “A neural-network approach for semiconductor wafer post-sawing inspection,” *IEEE Trans. Semicond. Manuf.*, vol. 15, no. 2, pp. 260–266, 2002.

[5] H. Lee, Y. Kim, and C. O. Kim, “A deep learning model for robust wafer fault monitoring with sensor measurement noise,” *IEEE Trans. Semicond. Manuf.*, vol. 30, no. 1, pp. 23–31, 2017.

[6] Y.-J. Cha, W. Choi, and O. Büyüköztürk, “Deep learning-based crack damage detection using convolutional neural networks,” *Comput. Civ. Infrastruct. Eng.*, vol. 32, no. 5, pp. 361–378, 2017.

[7] S. Zhou, Y. Chen, D. Zhang, J. Xie, and Y. Zhou, “Classification of surface defects on steel sheet using convolutional neural networks,” *Mater. Tehnol.*, vol. 51, no. 1, pp. 123–131, 2017.

[8] X. Tao *et al.*, “Wire defect recognition of spring-wire socket using multitask convolutional neural networks,” *IEEE Trans. Components, Packag. Manuf. Technol.*, vol. 8, no. 4, pp. 689–698, 2018.

[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. 6507, no. c, pp. 1–1, 2017.

[10] K. B. Lee and C. O. Kim, “Recurrent feature-incorporated convolutional neural network for virtual metrology of the chemical mechanical planarization process,” *J. Intell. Manuf.*, pp. 1–14, 2018.

[11] 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, p. 1, 2018.

[12] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” *Adv. Neural Inf. Process. Syst.*, vol. 19, no. 1, p. 153, 2007.

[13] C. M. Bishop, *Pattern recognition and machine learning*, no. 9. Springer, 2006.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[15] Y. LeCun, B. Yoshua, and H. Geoffrey, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[16] 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, pp. 1929–1958, 2014.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[18] C. F. Kuo, C. M. Hsu, C. Fang, S.-M. Chao, and Y.-D. Lin, “Automatic defect inspection system of colour filters using Taguchi-based neural network,” *Int. J. Prod. Res.*, vol. 51, no. 5, pp. 1464–1476, 2013.

[19] V. N. Vapnik, *Statistical learning theory*. Wiley, 1998.