An Improved Capsule Network (WaferCaps) for Wafer Bin Map Classification Based on DCGAN Data Upsampling

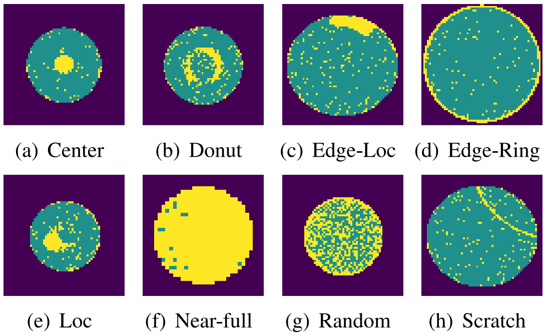
Abd Al Rahman M. Abu Ebayyeh, Sebelan Danishvar, and Alireza Mousavi, *Senior Member, IEEE*

***Abstract*—Wafer bin maps contain vital information that helps semiconductor manufacturers to identify the root causes and defect pattern failures in wafers. Conventional manual inspection techniques in inspecting these failures are labour intensive and cause prolonged production cycle time. Therefore, automatic inspection techniques can solve this problem. This paper proposes a deep learning approach based on deep convolutional generative adversarial network (DCGAN) and a new Capsule Network (WaferCaps). DCGAN was used to upsample the original dataset and therefore increase the data used for training and balance the classes at the same time. While WaferCaps was proposed to classify the defect patterns according to eight classes. The performance of our proposed DCGAN and WaferCaps was compared with different deep learning models such as the original Capsule Network (CapsNet), CNN, and MLP. In all of our experiment, WM-811K dataset was used for the data upsampling and training. The proposed approach has shown an effective performance in generating new synthetic data and classify them with training accuracy of 99.59%, validation accuracy of 97.53% and test accuracy of 91.4%.**

***Index Terms*—Capsule network, data augmentation, deep learning, defect detection, generative adversarial network (GAN), pattern recognition, semiconductor manufacturing, wafer bin map (WBM).**

I. Introduction

ADVANCES in semiconductor technology and design have been the driving forces behind the successful progress of microelectronic and optoelectronic devices. The majority of these devices are manufactured using semiconductor wafers that consist of several hundreds of integrated circuits (ICs) (also called dies) [1], [2]. However, the fabrication process for the semiconductor wafers is complex and consist of many stages that should take place in a clean room environment, such as oxidation, photolithography, etching, ion implementation, and metallization, which requires monitoring many key process parameters. The complexity of these steps makes the wafer prone to many kinds of defects and failures; therefore, wafer testing is an essential step in order to provide necessary information on specific manufacturing problems, which can then reduce products' flaws and lead to early prevention [3]. One of the testing methods used in inspecting the wafer is called circuit probe in which each die in the wafer is tested using an electrical probe to form a binary map image called 'Wafer Bin Map' (WBM). In WBM, the defective dies are represented with logic '1' and the normal dies with logic '0'.


(a) Center (b) Donut (c) Edge-Loc (d) Edge-Ring

(e) Loc (f) Near-full (g) Random (h) Scratch

Fig. 1. Different WBM patterns (yellow dots indicate defective dies).

Defective dies in the wafer tend to form a spatial pattern [4]. Figure 1 shows common inspected defect patterns in WBMs. An experienced inspector can identify the cause of defect depending on the WBM's pattern.

The process of manually inspecting these defects is time consuming and may be affected by the fatigue's level of the inspector, especially because modern semiconductors manufacturers produce several thousands of wafers every week [5], [6]. Therefore, many semiconductor manufacturing are investigating this problem using machine learning and computer vision techniques to perform automatic defect detection [3].

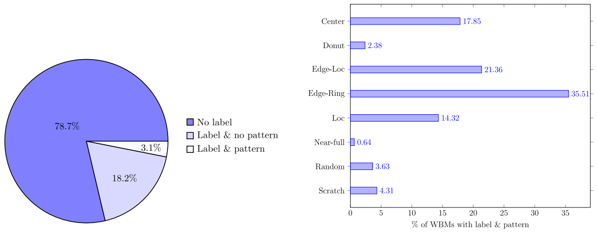
In this paper, we use deep learning models to solve the WBM classification problem automatically. We investigated WM-811K dataset that consists of 811,457 WBMs. In total, 21.3% of the WBMs in the dataset have labels while the rest do not have. Among labeled WBMs, 3.1% have failure patterns while 18.2% do not have patterns. Figure 2(a) summarizes the distribution of data for WM-811k dataset. In this article, we considered labelled and patterned data that account for 3.1% of the total. It is clear that the labeled and patterned WBMs data considered are highly imbalanced as shown in Figure 2(b). Furthermore, the number of images of near-full class is very low compared with other classes. Data imbalance problem

Manuscript received September 23, 2021; accepted December 8, 2021. Date of publication December 10, 2021; date of current version February 3, 2022. This work was supported by IQONIC Project through the European Union's Horizon 2020 Research and Innovation Program under Grant 820677. *(Corresponding author: Abd Al Rahman M. Abu Ebayyeh.)*

The authors are with the Department of Electronic and Computer Engineering, Brunel University London, Uxbridge UB8 3PH, U.K. (e-mail: abdalrahman.abuebayyeh@brunel.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSM.2021.3134625.

Digital Object Identifier 10.1109/TSM.2021.3134625



(a) Data distribution (b) Data with label & pattern.

Fig. 2. WM-811k data availability.

plays negative role in the overall classification performance as it will be biased towards the majority class of the data [7], [8].

Motivated by the previous reasons, we used Deep Convolutional Generative Adversarial Network (DCGAN) to balance the dataset and increase them at the same time. Then we used a newly proposed Wafer Capsule Network (WaferCaps) for classification and compared the accuracy with different deep learning models. To the best of our knowledge, this is the first paper that uses capsule network for WBM classification. In summary, this paper is organized as follows. In Section II, we will discuss the related work to our research. In Section III, we will discuss the methodology in analyzing the dataset, DCGAN upsampling process and we will introduce our proposed WaferCaps. In Sections IV and V, we will present our results, discuss them, and compare our main algorithm with different deep learning models. Our article will finally conclude in Section VI.

II. Related Work

In recent years, many research articles conducted machine learning techniques (both supervised and unsupervised) to identify WBM defect patterns [9]. Most of the unsupervised techniques such as clustering were used to identify and categorize the defect patterns into new categories. While supervised techniques such as Support Vector Machines (SVM), Decision Trees (DT) were mainly used for detection and classification.

In regards to unsupervised learning methods, Yuan and Kuo [10], [11] used model-based clustering techniques where they combined spatial non-homogeneous Poisson distribution, binary normal distribution, and principal curve model for further identification of various amorphous/linear and curvilinear WBM defect patterns. However, their approach is mainly based on simulated results and lacks the capability to detect commonly investigated WBM patterns in literature [1]. Wang [12] used Support Vector Clustering (SVC) to identify different types of mixed WBM defect patterns such as multiple zones, multiple scratches, multiple rings and ring-zone. However, their algorithm is computationally expensive due to the large number of clusters chosen [1]. Jin *et al.* [4] proposed a clustering-based algorithm namely density-based spatial clustering of applications with noise (DBSCAN); the proposed approach was used for outlier detection and defect cluster pattern extraction. With this procedure, arbitrarily shaped cluster patterns can be detected without specifying the number of clusters in advance. Park *et al.* [13] proposed a class label reconstruction method based on clustering for subdividing a defect class with various patterns into several groups, creating a new class for defect samples that cannot be categorized into known classes and detecting unknown defects. However, experimental results demonstrated that the number of clusters was too large compared to the number of classes.

Regarding supervised learning methods, Liao *et al.* [14] proposed a WBM defect pattern classification method based on morphology similarity approach and SVM. Morphology similarity was first used to generate single and mixed type patterns with certain degrees of similarity, as compared to the objective target WBMs. SVM classifier is then conducted to classify the generated patterns according to the class being specified. The proposed method showed that an overall catching rate of 95% with only 5% false-alarm rate had been achieved. However, their approach showed low performance in detecting donut and repeated scratches patterns, where 72% and 73% catching rates were achieved for detecting these patterns, respectively. Piao *et al.* [15] proposed DT ensemble-based WBM failure pattern recognition method based on the radon transform features. The radon transform were used as a mean to extract four features to be used as classifier's input. The DT classifier classified the defect pattern according to eight failure patterns. The accuracy of the classifier achieved 90.5% classification accuracy for all the selected patterns; however, the proposed method failed to recognize several pattern types efficiently.

More recently, deep learning has been widely used for machine vision, pattern recognition and automatic visual inspection problems. Deep learning is applied by using three or more hidden layers in artificial neural network (ANN) structure. Convolutional Neural Networks (CNNs) and Autoencoders are common deep learning algorithms used in literature. These networks are specialized in machine vision problems due to their image handling properties in extracting image features and classify the images according to certain categories (classes) [1]. Kyeong and Kim [16], proposed CNN algorithm to classify mixed defects in WBM. They used four individual CNNs for each defect pattern, such that each CNN apply binary classification to determine whether a corresponding pattern exists when several defect patterns are mixed over a wafer. However, to detect a mixed pattern defect, the output of each individual classifier must be obtained. Yu [2] proposed an enhanced stacked denoising autoencoder (ESDAE) algorithm to detect WBM defects. They have also used manifold regularization in the learning procedure, which improves the algorithm's performance effectively due to the preservation of intrinsic information in the data. The overall detection accuracy of the proposed method reached 89.6%. A similar approach was also used in [17] where they combined SDAE and CNN to form a new deep learning model called stacked convolutional sparse denoising auto-encoder (SCSDAE). The new combined model was able to learn effective features and accumulate the robustness layer by layer, which adopts SDAE as the feature extractor and stacks well- designed fully connected SDAE in a convolutional way to obtain much robust feature representations. However, the previous two studies faced two major problems; the lack of enough data for training and data imbalance. For instance, in [17], the accuracy of identifying near-full class was 87.5% only based on 54 images used in training.

III. Methodology

Our proposed method consists of two main algorithms; DCGAN for generating new WBM samples and CapsuleNet for classifying the total WBM samples. Since the WBMs that we are dealing with have only three intensity values; background, logic 1 (represent defective dies) and logic 0 (represent normal dies), there is no need to deal with RGB images and it will be sufficient to process the images in grayscale. Therefore, all WBMs are converted into grayscale before supplying them to DCGAN. All the WBM images in this paper were unified to a size of 64 × 64.

A. Data Upsampling Using DCGAN

As can be seen from Figure 2(b), the WM-811K dataset with pattern and label is considered highly imbalanced, for instance, the Edge-Ring class has 8268 sample images, whereas the number of samples in the Near-full class is 148 images only. Accordingly, some of the classes do not have sufficient dataset for the training process and will make the accuracy biased towards the dominant class. Therefore, upsampling is necessary to optimize the classification results and to ensure that the total accuracy is not biased to any class more than the other. In this paper, Deep Convolutional Generative Adversarial Network (DCGAN) were used for upsampling the dataset and for increasing the training data for better accuracy.

GAN was first introduced in 2014 by Goodfellow *et al.* [18]. It consists of two neural networks, namely, generator and discriminator. The original paper suggested that both generator and discriminator are multilayer perceptron networks (MLP). The generator is responsible for producing synthetic images that look like the training dataset and supply these produced images for the discriminator. The discriminator is responsible for making the decision whether the produced images from the generator look similar to the real data or not using binary classification scheme. This framework can be used to generate realistic new images that are almost identical to pre-existing training dataset by training the generator and discriminator simultaneously using adversarial process.

DCGAN proposed in 2016 by Radford *et al.* [19] and is considered an extension of the original GAN proposed by Goodfellow *et al.* [18]. The architecture of DCGAN is almost the same in the original GAN except that convolutional (conv) and conv transpose (deconv) layers are used in discriminator and generator networks respectively instead of the MLP structure. In our DCGAN, the generator network receives a one-dimensional random Gaussian vector of size 100 as an input. Multiple deconv layers are then applied to upscale the vector into 64×64 random noise image. All the deconv layers are followed by ReLU activation function and batch normalization was used in these layers in order to stabilize the learning process. The final layer will have the same size as the target image (64 × 64), which will be a noise image in the first epoch of training that will evolve in each epoch to produce the wanted synthetic image. The layers of the generator network are explained in table I.

TABLE I
Generator Network

Layer	Type	Input size	Kernel Size/ Stride	Activation	Batch Normalization	Output size
1	FC	[100,]	3/1	-	No	[16,16,128]
2	Deconv	[16,16,128]	3/2	ReLU	Yes	[32,32,64]
3	Deconv	[32,32,64]	3/1	ReLU	Yes	[32,32,32]
4	Deconv	[32,32,32]	3/2	ReLU	No	[64,64,1]

TABLE II
Discriminator Network

Layer	Type	Input size	Kernel Size/ Stride	Activation	Batch Normalization/ Dropout	Output size
1	Conv	[64,64,1]	3/2	L.ReLU	Dropout	[32,32,32]
2	Conv	[32,32,32]	3/2	L.ReLU	Batch N.	[16,16,64]
3	Conv	[16,16,64]	3/2	L.ReLU	Batch N.	[8,8,128]
4	FC	[8,8,128]	-	-	-	[8192,]
5	FC	[8192,]	-	Sigmoid	-	[1,]

The discriminator in DCGAN is simply a CNN that does binary classification; it receives the training wafer maps and label them real (class 1), while the output generated from the generator is labeled as fake (class 0). Each conv block in the discriminator is followed by LeakyReLU activation function with $\alpha = 0.2$. Dropout and batch normalization has been used in the first three conv layers to stabilize the learning process. The output layer in the discriminator uses a Sigmoid function for the classification process (Real or Fake). The layers of the discriminator network are explained in table II.

During training, the generator progressively becomes better at creating images that look real, while the discriminator becomes better at distinguishing real from synthetic images. The process reaches equilibrium when the discriminator can no longer distinguish between real and synthetic images. In this paper, we used the minimax concept suggested by Goodfellow *et al.* [18] to fulfill this objective, which is given by Equation (1). Equation (1) indicates that the discriminator is trained to maximize the probability $\log(D(x))$ of assigning the correct label to both the generated synthetic data probability and real training data using while the generator is trained to minimize $\log(1 - (G(z)))$.

$$\min_{G}\max_{D} V(D, G) = \mathbb{E}_{x\sim p_{data}(x)}\big[\log(D(x))\big] + \mathbb{E}_{z\sim p_z(z)}\big[\log(1 - (G(z)))\big] \quad (1)$$

where $D(x)$ is the probability that x belong to the original data distribution, $G(z)$ is the generator function that maps to the data space, $\mathbb{E}_{x\sim p_{data}(x)}$ is the expected value over all real samples, $\mathbb{E}_{z\sim p_z(z)}$ is the expected value over all fake samples [20].

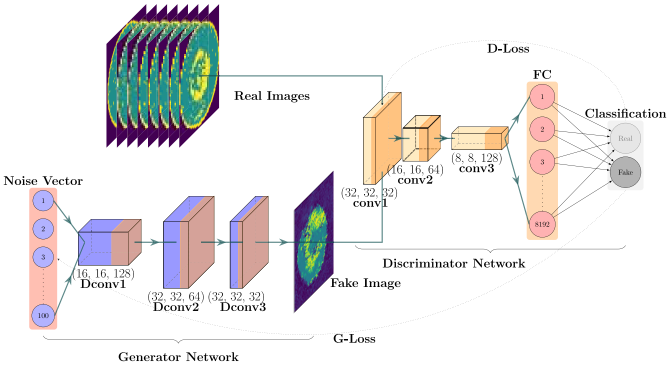


Fig. 3. Architecture of the proposed DCGAN for generating synthetic WBMs.

Since discriminator is performing binary classification (real or fake), binary cross entropy (BCE) loss function (given by equation (2)) was used for the discriminator to penalize itself for misclassifying real and fake images.

$$J_q(w) = -\frac{1}{N}\sum_{n=1}^{N} y_n \cdot \log(q(y_n)) + (1-y_n)\cdot \log(1-q(y_n))) \tag{2}$$

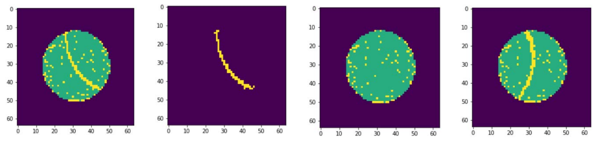
where y_n is the label for training example n (1 for real samples and 0 for fake), $q(y)$ is the predicted probability of the point being real for all N points. The architecture of DCGAN used in this research can be summarized in Figure 3.

A post-processing step using Pearson correlation coefficients (PCC) is employed to determine whether to accept the DCGAN generated images or dismiss them. Pearson's method is widely used in statistical analysis, pattern recognition and applications that require matching and comparing two images as described in equation (3) [21].

$$r_1 = \frac{\sum_i (x_i - x_m)\cdot(y_i - y_m)}{\sqrt{\sum_i (x_i - x_m)^2}\cdot\sqrt{\sum_i (y_i - y_m)^2}} \tag{3}$$

where x_i is the intensity of the i^{th} pixel in image A, y_i is the intensity of the i^{th} pixel in image B. x_m is the mean intensity of image A, and y_m is the mean intensity of image B. The absolute values of PCC are between 0 and 1. They show how much two images are similar. The closer the coefficient to 1, the more the two images are similar to each other. Based on piratical experiment, a threshold of 0.92 were used for comparison, where the output of the DCGAN networks is compared with all the database images of the class using Pearson's method. If the similarity of the DCGAN generated image is higher than this threshold, the image will be considered; otherwise, it will be ignored. Experiments show that considering this threshold, almost 50% of DCGANs output are accepted and added to the database. The accepted dataset are then used as training set for our proposed WaferCaps to perform classification.

Using this method, we were able to generate the required synthetic images for training our proposed WaferCaps. However, our DCGAN could not generate synthetic WBMs for the scratch defect pattern. Therefore, we used different approach for upsampling the WBMs of this class. Our approach proposed isolating the scratch pattern from each WBM and apply rotation operations of the pattern on the WBM without the pattern. Figure 4 demonstrates applying this approach on one of the WBM samples that contain scratch defect pattern.



(a) Original (b) Only scratch (c) Background (d) New Sample

Fig. 4. Image upsampling procedure for Scratch class.

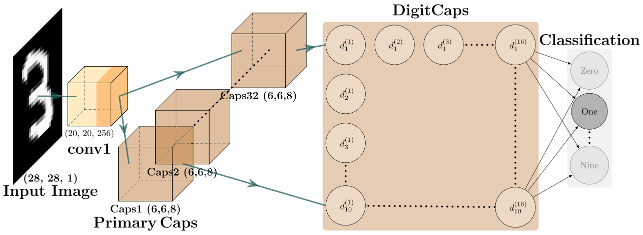


Fig. 5. Originally proposed CapsNet for MNIST handwritten digit classification [24].

B. *WaferCaps*

CNNs have been long a popular deep learning tool in dealing with machine vision problems [22]. Despite their remarkable performance in image classification tasks, CNNs have several drawbacks. For instance, the pooling layers in the CNN can decrease the number of features extracted by the network and therefore valuable information in the image will be lost. Furthermore, CNNs are not very good at detecting the spatial location of the features in the image [23]. CapsNet is a newly proposed neural network that can overcome the previous problems. It was originally proposed by Sabour *et al.* in 2017 [24] to classify MNIST handwritten digits.

Two key aspects distinguish CapsNets from CNNs, which are layer-based squashing and dynamic routing [25]. CapsNet replaces the scalar-output feature detectors of CNNs with vector-output capsules and replaces pooling with routing-by-agreement. Each neuron in the capsule represents various features in particular parts of an image. In this way, the whole entity of the image can be recognized by considering each part [26]. The architecture of the original CapsNet proposed in Sabour *et al.* paper [24] is shown in Figure 5. As we can see in Figure 5, the original network is made up of four major layers [27].

- Conv layer, which is a standard conv layer as the one found in CNNs. The input image size for this layer is 28 × 28. 265 kernels were used along with stride of 1 and ReLU activation function and to generate the feature maps and each kernel has a size of 9 × 9. The resulting feature map output after applying the layer has a size of 20 × 20 × 265.
- PrimaryCaps. In this layer, the feature maps resulted from the conv layer were split into 32 capsules where conv operations are performed. The kernel's size in this layer is again 9 × 9 and a stride of 2. This will produce an output size of 6 × 6 × 8 for each capsule.

- DigitCaps. This layer has 16 capsules per digit class and each of these capsules receives input from all the capsules in the previous layer.
- Fully connected layer for classification. The input image size was 28 × 28 to match the size of MNIST data used for evaluating the network.

Else than the first layer that shares some similarities with CNN in the feature extraction process, the other layers behave in a different manner. In the second layer (PrimaryCaps), each capsule i from the 32 has an activity vector $\mathbf{u}_i$ to encode the spatial information in the form of instantiation parameters. Then the output of $\mathbf{u}_i$ is fed to the next layer (DigitCaps), such that each capsule j from the 16 per digit class will receive $\mathbf{u}_i$ and multiply with the weight matrix $\mathbf{W}_{ij}$. This will result in the prediction vector $\hat{\mathbf{u}}_{j|i}$, which indicates the amount of contribution for capsule i in the PrimaryCaps on capsule j in the DigitCaps as given by equation (4).

$$\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i \tag{4}$$

The predictions are then multiplied by a coefficient called coupling coefficient c that represents the agreement between capsules. Hence, coefficient c is updated based on iterative process to form what so called "Dynamic Routing". This process can be determined by routing softmax function whose initial logits b_{ij} are the log prior probabilities that capsule i in the PrimaryCaps should be coupled to capsule j in the DigitCaps. This operation can be demonstrated by equations (5)-(8).

$$a_{ij} = \mathbf{s}_j \cdot \hat{\mathbf{u}}_{j|i} \tag{5}$$

$$b_{ij} = b_{ij} + a_{ij} \tag{6}$$

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ij})} \tag{7}$$

$$\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i} \tag{8}$$

where $\mathbf{s}_j$ is weighted sum that is calculated to obtain the candidates for a squashing function v_j. The squashing operation is responsible for creating a normalized vector from the multiple neurons contained in the capsule. The activation function used in this step is given by equation (9).

$$v_j = \frac{\|\mathbf{s}_j\|^2}{1 + \|\mathbf{s}_j\|^2} \cdot \frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}. \tag{9}$$

A margin loss function was defined to assist the classification process. The function evaluates the loss term coming from the output vector of DigitCaps. This will help in deciding whether the selected digit capsule matches the real target value of class k. The formula of the margin loss function is given by equation (10).

$$L_k = T_k \max\left(0, m^+ - \|v_k\|\right)^2 + \lambda(1 - T_k)\max\left(0, \|v_k\| - m^-\right)^2 \tag{10}$$

where T_k is a label (0 or 1) indicating whether a class k is present '1' or not '0'. Terms m^+ , m^-, and λ are the hyper-parameters of the model such that $m^+ = 0.9$, $m^- = 0.1$ and $\lambda = 0.5$.

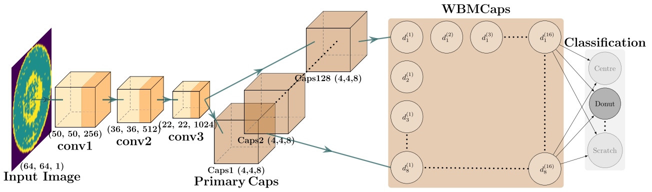


Fig. 6. Our proposed WaferCaps architecture.

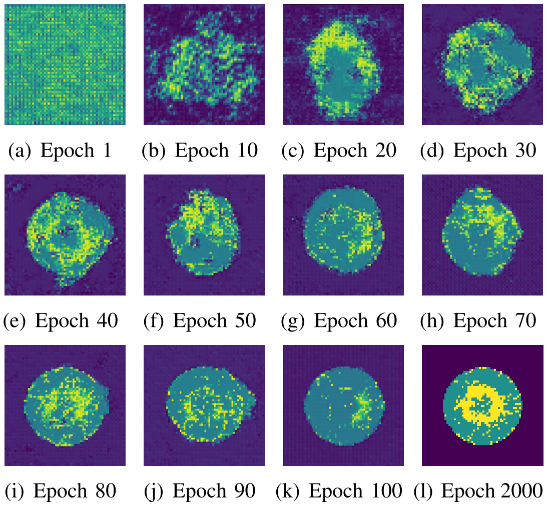
TABLE III
LAYERS OF THE PRPOSED WAFERCAPS

Layer	Type	Input size	Kernel Size/ Stride	Activation	Dropout	Output size
1	conv1	[64,64,1]	15/1	ReLU	Yes	[50,50,256]
2	conv2	[50,50,256]	15/1	ReLU	Yes	[36,36,512]
3	conv3	[36,36,512]	15/1	ReLU	Yes	[22,22,1024]
4	PrimCaps	[22,22,1024]	9/2	ReLU	No	[4,4,8,128]
5	WBMCaps	[4,4,8,128]	-	Squash	No	[16,8]
6	FC	[16,8]	-	Softmax	No	[8,]

Despite the effciency of the original CapsNet in dealing with MNIST handwritten digits classification [18], original architecture of CapsNet has some drawbacks that we must modify to fit our dataset. The original architecture employs two conv layers to extract image features, which is not proper for complex images. Furthermore, the size of the conv kernels in the original CapsNet is 9 × 9 that compatible with the MNIST dataset (28 × 28). For datasets with big images size, this kernel produces a large number of training parameters. In this paper, we propose a new Wafer Capsule Network (WaferCaps) to overcome the limitations of the original CapsNet. To intensify the capability of conv layers to extract image features, we add two more conv layers and establish a dropout layer to avoid overfitting after each layer. The input size of the network has also been modified to 64 × 64 to match our WBMs size. The architecture of the WaferCaps we propose is shown in Figure 6. Also, Table III provides the layers details used in the architecture. As Figure 6 shows, conv1, conv2 and conv3 have 256, 512 and 1024 depth layers, respectively. All conv layers have 15 × 15 convolution kernels with a stride of 1 and ReLU activation. These layers transform pixel intensities to local feature detectors' activities fed as inputs to the primary capsules. The PrimaryCaps layer is a conv capsule layer with 128 channels of conv capsules. Each primary capsule comprises eight conv units with a 9 × 9 kernel and a stride of 2.

IV. Experimental Results

In this section, we implemented different sets of experiments to study the evaluate the performance of our proposed WaferCaps & DCGAN method. This study adopts accuracy, recall, precision and F1-score metrics to evaluate the performances of these methods. The accuracy of the algorithm predicts the number of images is classified correctly. The precision measures the exact efficiency of the algorithm for predicting the positive samples. The recall is the measure to calculate the true positive, and the mean harmonic of the recall


(a) Epoch 1 (b) Epoch 10 (c) Epoch 20 (d) Epoch 30

(e) Epoch 40 (f) Epoch 50 (g) Epoch 60 (h) Epoch 70

(i) Epoch 80 (j) Epoch 90 (k) Epoch 100 (l) Epoch 2000

Fig. 7. Generated synthetic WBMs for DCGAN over multiple epochs.

and precision is determined with F1 score. For calculating the four performance measurements, according to the confusion matrix, the values of true positive (TP), true negative (TN), false negative (FN) and false positive (FP) can be achieved. By calculating the above four values, accuracy, precision, recall, and F1 score are obtained based on equations (11)-(14).

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{11}$$

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

$$F1 = \frac{2(Precision \times Recall)}{Precision + Recall}. \tag{14}$$

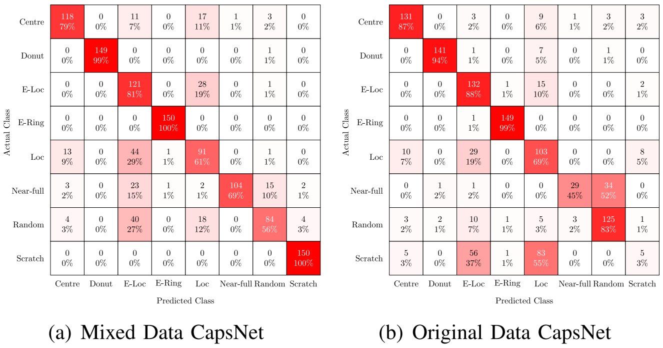
A. DCGAN

The training of DCGAN is performed over two steps. First step is training the discriminator alone on fake and real data, such that the discriminator can classify them efficiently. In this step, the generator is standby and is not being trained yet. The discriminator loss penalizes the discriminator for misclassifying a real sample as fake or a fake sample as real while updating the weights via backpropagation. The second step involves training the generator to create synthetic images such that the generator's loss penalizes the generator for producing a sample that the discriminator network classifies as fake. In our study, we used DCGAN on each class of the eight to generate synthetic data.

We upsampled all the classes such that each class will contain 10,000 samples (including original samples). The training for each class procedure involved 2,000 epochs. We started to get acceptable results after epoch 100 approximately and we got almost identical results after epoch 1,000. Figure 7 shows how the results are improved as the number of epochs is increasing for the donut class.

B. Experimental Data

The dimension of each WBM in the sample data is 64 × 64, and consists of eight labels: center, donut, edge-loc, edge-ring, loc, near-full, random, and scratch (shown in Figure 1). We divided the WBM data into two sets in order to verify and compare our methods; the first set is the original labelled and patterned data in WM-811K dataset, we called this dataset *original dataset*. The total WBMs used for training are 17,804, for validation 4,333 and for testing 2,165. The second set is a combination of the original data and the DCGAN generated WBMs, we called this dataset *mixed dataset*. We made sure in both groups that the WBMs used for testing are the same and are all of the original dataset. The total WBMs used for training are 63,200 and for validation are 15,600. Using these groups will allow us to explore the influence of using DCGAN generated data on the testing accuracy.



(a) Mixed Data CapsNet (b) Original Data CapsNet

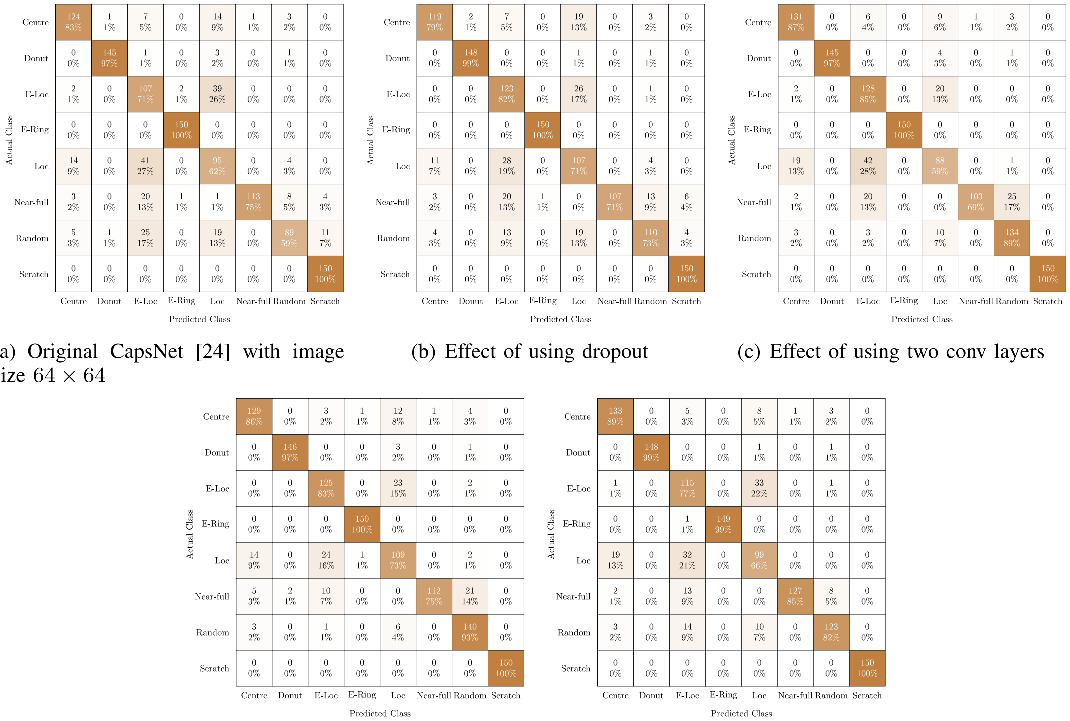
Fig. 8. Confusion matrix for test data according to original CapsNet [24] (image size of 28 × 28).

C. Ablation Study: Parameter Impact on CapsNet

Multiple parameters in deep learning models can present a significance difference in the model's performance. We varied all the possible parameters of the original CapsNet in order to observe the effect of that and come up with our proposed WaferCaps. Some of these parameters had minor noticeable effect on our model while varying others such as image size, dropout, number of conv layers, and kernel size demonstrated significant improvement on the model's overall accuracy. Therefore, in this study, we present those parameters that highly affected our model in accordance with the test accuracy and chose an optimal configuration that can meet the best test accuracy.

1) The Impact of Image Size: Similar to CNNs, capsule neural networks receive one fixed size of an image for all the samples supplied for training. The larger the fixed size, the less shrinking required and therefore the less deformation of features and patterns inside the image [28]. In this section we will investigate the originally proposed CapsNet by Sabour *et al.* [24] for WBM of size 28 × 28 (similar to the size of MNIST handwritten digits data they used) and compare the performance of this size with WBM of size 64 × 64. The comparison shows that by considering a size of 28 × 28, we get a test accuracy of 80.6%, while using an image size of 64 × 64 improved the test accuracy to be 82.9%. Figures 8(a) and 9(a) represent the confusion matrices for test data upon using image sizes of 28 × 28 and 64 × 64 respectively on original proposed CapsNet [24].

2) The Impact of Dropout: Dropout is widely conducted in the training of deep learning models as an effective



	Centre	Donut	E-Loc	E-Ring	Loc	Near-full	Random	Scratch
Centre	124 83%	1 1%	7 5%	0 0%	14 9%	1 1%	3 2%	0 0%
Donut	0 0%	145 97%	1 1%	0 0%	3 2%	0 0%	1 1%	0 0%
E-Loc	2 1%	0 0%	107 71%	2 1%	39 26%	0 0%	0 0%	0 0%
E-Ring	0 0%	0 0%	0 0%	150 100%	0 0%	0 0%	0 0%	0 0%
Loc	14 9%	0 0%	41 27%	0 0%	95 62%	0 0%	4 3%	0 0%
Near-full	3 2%	0 0%	20 13%	1 1%	1 1%	113 75%	8 5%	4 3%
Random	5 3%	1 1%	25 17%	0 0%	19 13%	0 0%	89 59%	11 7%
Scratch	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	150 100%

(a) Original CapsNet [24] with image size 64 × 64

	Centre	Donut	E-Loc	E-Ring	Loc	Near-full	Random	Scratch
Centre	119 79%	2 1%	7 5%	0 0%	19 13%	0 0%	3 2%	0 0%
Donut	0 0%	148 99%	0 0%	0 0%	1 1%	0 0%	1 1%	0 0%
E-Loc	0 0%	0 0%	123 82%	0 0%	26 17%	0 0%	1 1%	0 0%
E-Ring	0 0%	0 0%	0 0%	150 100%	0 0%	0 0%	0 0%	0 0%
Loc	11 7%	0 0%	28 19%	0 0%	107 71%	0 0%	4 3%	0 0%
Near-full	3 2%	0 0%	20 13%	1 1%	0 0%	107 71%	13 9%	6 4%
Random	4 3%	0 0%	13 9%	0 0%	19 13%	0 0%	110 73%	4 3%
Scratch	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	150 100%

(b) Effect of using dropout

	Centre	Donut	E-Loc	E-Ring	Loc	Near-full	Random	Scratch
Centre	131 87%	0 0%	6 4%	0 0%	9 6%	1 1%	3 2%	0 0%
Donut	0 0%	145 97%	0 0%	0 0%	4 3%	0 0%	1 1%	0 0%
E-Loc	2 1%	0 0%	128 85%	0 0%	20 13%	0 0%	0 0%	0 0%
E-Ring	0 0%	0 0%	0 0%	150 100%	0 0%	0 0%	0 0%	0 0%
Loc	19 13%	0 0%	42 28%	0 0%	88 59%	0 0%	1 1%	0 0%
Near-full	2 1%	0 0%	20 13%	0 0%	0 0%	103 69%	25 17%	0 0%
Random	3 2%	0 0%	3 2%	0 0%	10 7%	0 0%	134 89%	0 0%
Scratch	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	150 100%

(c) Effect of using two conv layers

	Centre	Donut	E-Loc	E-Ring	Loc	Near-full	Random	Scratch
Centre	129 86%	0 0%	3 2%	1 1%	12 8%	1 1%	4 3%	0 0%
Donut	0 0%	146 97%	0 0%	0 0%	3 2%	0 0%	1 1%	0 0%
E-Loc	0 0%	0 0%	125 83%	0 0%	23 15%	0 0%	2 1%	0 0%
E-Ring	0 0%	0 0%	0 0%	150 100%	0 0%	0 0%	0 0%	0 0%
Loc	14 9%	0 0%	24 16%	1 1%	109 73%	0 0%	2 1%	0 0%
Near-full	5 3%	2 1%	10 7%	0 0%	0 0%	112 75%	21 14%	0 0%
Random	3 2%	0 0%	1 1%	0 0%	6 4%	0 0%	140 93%	0 0%
Scratch	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	150 100%

(d) Effect of using three conv layers

	Centre	Donut	E-Loc	E-Ring	Loc	Near-full	Random	Scratch
Centre	133 89%	0 0%	5 3%	0 0%	8 5%	1 1%	3 2%	0 0%
Donut	0 0%	148 99%	0 0%	0 0%	1 1%	0 0%	1 1%	0 0%
E-Loc	1 1%	0 0%	115 77%	0 0%	33 22%	0 0%	1 1%	0 0%
E-Ring	0 0%	0 0%	1 1%	149 99%	0 0%	0 0%	0 0%	0 0%
Loc	19 13%	0 0%	32 21%	0 0%	99 66%	0 0%	0 0%	0 0%
Near-full	2 1%	0 0%	13 9%	0 0%	0 0%	127 85%	8 5%	0 0%
Random	3 2%	0 0%	14 9%	0 0%	10 7%	0 0%	123 82%	0 0%
Scratch	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	150 100%

(e) Effect of using a kernel size of 7 × 7

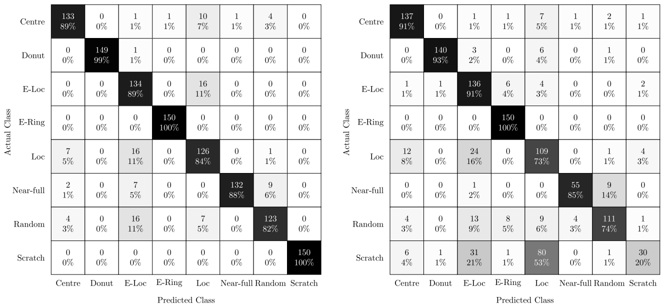
Fig. 9. Confusion matrices for the test data with different parameter scenarios.

regularization and implicit model ensemble technique [29]. In this section, we will choose the best CapsNet from Section IV-C1 (image size 64 × 64) and investigate the effect of applying dropout on the test accuracy. Hence, the dropout will be applied on the conv layer before the PrimaryCaps with a percentage of 50%. Figure 9(b) show the confusion matrix for the test data with applying dropout. We can observe from Figure 9(b) that the test accuracy achieved was 84.5% when dropout was used, which is better by 1.6% compared with no dropout.

3) The Impact of Number of Layers: Having successive conv networks in deep learning model will contribute to the learning process of the model in identifying complex and important features. Therefore, the number of layers will highly affect the outcomes of classification. As stated in Section III-B, the original CapNet has only one conv layer. In this section, we will modify the best CapsNet from Section IV-C2 (with dropout) and alternate the number of conv layers to be two and three layers. We noticed from the experiments that the test accuracy for two and three conv layers were 85.8% and 88.4% respectively. Figures 9(c) and 9(d) demonstrate the confusion matrices after applying these updates. Therefore, we concluded that using three conv layer resulted in better test accuracy.

4) The Impact of Kernel Size: Due to the working principle of conv layers, all the kernels are sliding on the image. Depending on kernel size, each successive layer will have different feature map size and depth according to equation (15).

$$n_{out} = \left(\frac{n_{in} + 2p - k}{s}\right) + 1 \tag{15}$$



	Centre	Donut	E-Loc	E-Ring	Loc	Near-full	Random	Scratch
Centre	133 89%	0 0%	1 1%	1 1%	10 7%	1 1%	4 3%	0 0%
Donut	0 0%	149 99%	1 1%	0 0%	0 0%	0 0%	0 0%	0 0%
E-Loc	0 0%	0 0%	134 89%	0 0%	16 11%	0 0%	0 0%	0 0%
E-Ring	0 0%	0 0%	0 0%	150 100%	0 0%	0 0%	0 0%	0 0%
Loc	7 5%	0 0%	16 11%	0 0%	126 84%	0 0%	1 1%	0 0%
Near-full	2 1%	0 0%	7 5%	0 0%	0 0%	132 88%	9 6%	0 0%
Random	4 3%	0 0%	16 11%	0 0%	7 5%	0 0%	123 82%	0 0%
Scratch	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	0 0%	150 100%

(a) Mixed Data proposed WaferCaps

	Centre	Donut	E-Loc	E-Ring	Loc	Near-full	Random	Scratch
Centre	137 91%	0 0%	1 1%	1 1%	7 5%	1 1%	2 1%	1 1%
Donut	0 0%	140 93%	3 2%	0 0%	6 4%	0 0%	1 1%	0 0%
E-Loc	1 1%	1 1%	136 91%	6 4%	4 3%	0 0%	0 0%	2 1%
E-Ring	0 0%	0 0%	0 0%	150 100%	0 0%	0 0%	0 0%	0 0%
Loc	12 8%	0 0%	24 16%	0 0%	109 73%	0 0%	1 1%	4 3%
Near-full	0 0%	0 0%	1 2%	0 0%	0 0%	55 85%	9 14%	0 0%
Random	4 3%	0 0%	13 9%	8 5%	9 6%	4 3%	111 74%	1 1%
Scratch	6 4%	1 1%	31 21%	1 1%	80 53%	0 0%	1 1%	30 20%

(b) Original Data proposed WaferCaps

Fig. 10. Confusion matrices for the test data with proposed WaferCaps.

where n_{out} is the resulting size of feature map, n_{in} is the input feature map to the layer, p is pooling size, k is the kernel size and s is the stride. Large kernel size will result in deeper conv layers and vice-versa. The original kernel size proposed by Sabour *et al.* [24] is 9 × 9 as explained in Section III-B. In this section, we will use the best CapsNet from Section IV-C3 and alternate the kernel size into other suitable sizes such as 7 × 7 and 15 × 15. Our experiments showed that using a kernel size of 7 × 7 resulted in a test accuracy of 87% which is less than using a kernel size of 9 × 9 as shown in previous section. However, using a kernel size of 15 × 15 improved the test accuracy to 91.4%. Figures 9(e) and 10(a) demonstrate confusion matrices of the test data after changing the kernel size to 7 × 7 and 15 × 15 respectively.

Kernel size is the last parameter we varied in this study, and by applying it we reach to our proposed WaferCaps that is described in Figure 6 and Table III.

TABLE IV
OVERALL TRAINING, VALIDATION AND TEST ACCURACIES USING MIXED DATASET

Model	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
WaferCaps (proposed)	**99.59%**	**97.53%**	**91.4%**
CapsNet [24]	99.9%	95.48%	80.6%
CNN	93%	92.7%	82.1%
MLP	96.8%	92.2%	76%

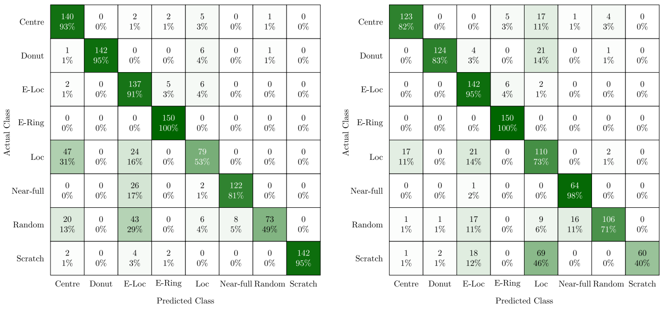
TABLE V
OVERALL TRAINING, VALIDATION AND TEST ACCURACIES USING ORIGINAL DATASET

Model	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**
WaferCaps (proposed)	**99.89%**	**88.92%**	**78.2%**
CapsNet [24]	97.84%	81.12%	73.1%
CNN	90%	89.7%	78.8%
MLP	91.72%	75.17%	60.9%

5) The Impact of Data Size: In this section, we present the influence of the training dataset size on our proposed WaferCaps. We considered the original and mixed datasets for this comparison to investigate the effect of using the synthetic WBMs generated by DCGAN on the test accuracy of our proposed WaferCaps. By using original dataset only for training, the test accuracy drops dramatically to 78.2% from 91.4% with using mixed dataset. Figure 10 show the confusion matrices of test data for using mixed and original dataset. This result proves the necessity of using DCGAN for data upsampling to get better test results. Tables IV and V also effect of changing the training dataset on the training and validation accuracies.

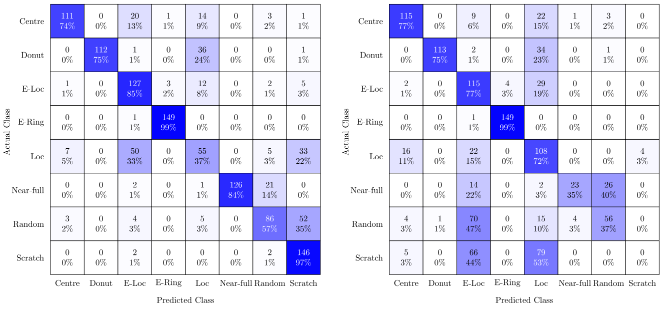
D. Comparison With Other Deep Learning Models

A series of experiments are conducted on both original and mixed dataset to compare our proposed WaferCaps method with several deep learning models, such as the original CapsNet [24], which we had to resize the WBM to a size of 28 × 28 to match the architecture of the CapsNet used. We have also used the CNN described in Figure 13, and the MLP described in Figure 14 for comparison. In CNN, the WBM images used for training are fed to the network in the same way as in CapNet and WaferCaps. However, in MLP, the training images were flattened, such that the input image of size 64 × 64 is transformed into a vector of size 4, 096 × 1 to be used in the input layer as shown in Figure 14. In this section, we have also demonstrated that all deep learning models perform better when they are trained with the mixed dataset rather than training them using original dataset only (see Tables IV and V). Figures 8, 10, 11 and 12 represent the confusion matrices of the test data for applying CapsNet [24], proposed WaferCaps, CNN and MLP respectively.



(a) Mixed Data CNN (b) Original Data CNN

Fig. 11. Confusion matrices for the test data with CNN.



(a) Mixed Data MLP (b) Original Data MLP

Fig. 12. Confusion matrices for the test data with MLP.

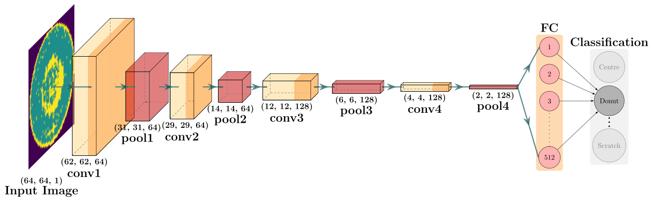


Fig. 13. CNN architecture used for comparison.

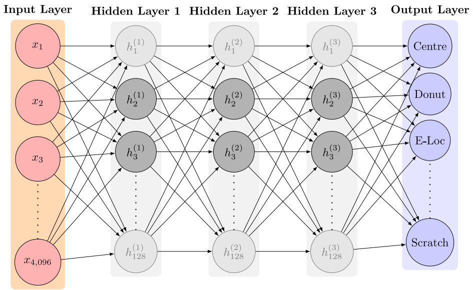


Fig. 14. MLP architecture used for comparison.

V. DISCUSSION & FUTURE WORK

We demonstrated the efficiency of using DCGAN for all the defect pattern in Sections III-A and IV-A. However, our DCGAN did not perform well when we tried to generate scratch defect patterns. In order to mitigate this issue we used the method described in Figure 4. Despite of the efficiency of this method, we could not generate more than 10,000 samples approximately due to the limitations of the number of scenarios that we can get. Therefore, we had to match this number of samples to the other patterns in order to have balanced

TABLE VI
METRICS FOR EVALUATING THE TEST DATA OF MIXED (DCGAN & ORIGINAL) DATASET FOR DIFFERENT DEEP LEARNING MODELS

Model	Metric	Center	Donut	E-Loc	E-Ring	Loc	N-full	Random	Scratch
WaferCaps (proposed)	**Recall**	**0.887**	**0.993**	**0.893**	**1.0**	**0.84**	**0.88**	**0.82**	**1.0**
	Precision	**0.911**	**1.0**	**0.766**	**0.993**	**0.792**	**0.992**	**0.898**	**1.0**
	F1-score	**0.899**	**0.997**	**0.825**	**0.997**	**0.816**	**0.933**	**0.857**	**1.0**
CapsNet [24]	Recall	0.787	0.993	0.807	1.0	0.607	0.693	0.56	1.0
	Precision	0.855	1.0	0.506	0.987	0.583	0.99	0.8	0.962
	F1-score	0.819	0.997	0.622	0.993	0.595	0.816	0.659	0.98
CNN	Recall	0.933	0.947	0.913	1.0	0.527	0.813	0.487	0.947
	Precision	0.660	1.0	0.581	0.943	0.76	0.94	0.973	1.0
	F1-score	0.773	0.973	0.71	0.971	0.622	0.871	0.65	0.973
MLP	Recall	0.74	0.747	0.847	0.993	0.367	0.84	0.573	0.973
	Precision	0.91	1.0	0.614	0.974	0.447	1.0	0.723	0.613
	F1-score	0.816	0.855	0.711	0.983	0.403	0.913	0.639	0.753

TABLE VII
METRICES FOR EVALUATING THE TEST DATA OF ORIGINAL ONLY DATASET FOR DIFFERENT DEEP LEARNING MODELS

Model	Metric	Center	Donut	E-Loc	E-Ring	Loc	N-full	Random	Scratch
WaferCaps (proposed)	**Recall**	**0.887**	**0.907**	**0.927**	**1.0**	**0.773**	**0.646**	**0.773**	**0.267**
	Precision	**0.875**	**1.0**	**0.698**	**0.993**	**0.483**	**0.875**	**0.841**	**0.784**
	F1-score	**0.881**	**0.951**	**0.797**	**0.997**	**0.595**	**0.743**	**0.806**	**0.398**
CapsNet [24]	Recall	0.873	0.94	0.88	0.993	0.687	0.446	0.833	0.033
	Precision	0.879	0.979	0.567	0.98	0.464	0.879	0.767	0.363
	F1-score	0.876	0.959	0.689	0.987	0.554	0.592	0.799	0.059
CNN	Recall	0.82	0.827	0.947	1.0	0.733	0.985	0.707	0.4
	Precision	0.866	0.976	0.7	0.932	0.482	0.79	0.938	1.0
	F1-score	0.842	0.895	0.805	0.965	0.582	0.877	0.806	0.571
MLP	Recall	0.767	0.753	0.767	0.993	0.72	0.354	0.373	0
	Precision	0.81	0.991	0.385	0.974	0.374	0.821	0.651	0
	F1-score	0.788	0.856	0.512	0.983	0.492	0.495	0.475	0

data for all the eight classes. In our upcoming research, we are planning to optimize DCGAN in order to generate scratch defect patterns so we can generate unlimited number of data for all the classes and observe the accuracy of classification according to that.

The results obtained in Section IV showed that our proposed WaferCaps performed the best in terms of test accuracy when mixed dataset was used for training instead of the original dataset only. The same was also observed when other deep learning models were used such as CapNet [24], CNN, and MLP.

However, from Table V, we can observe that CNN has slightly outperformed our proposed WaferCaps when original dataset used only for training. Table IV also shows that CapNet [24] performed better than our proposed WaferCaps in terms of the training accuracy; however, the better result in training accuracy is due to overfitting as both validation and test accuracies scored better when WaferCaps was used. Hence, the Dropout method was used to prevent overfitting.

Noteworthy, in Section IV-C that changing certain parameters such as image size, number of conv layers, and kernel size can affect the test accuracy when both datasets used (mixed and original).

From the confusion matrices in Figures 8, 9,11 and 12, we can observe that most of the defect patterns such as Donut, Edge-Ring and Scratch were very easy to detect when mixed dataset are used for training. While we noticed that it becomes much harder to detect some classes such as Scratch when original dataset was used in training. This conclusion could lead to future work, where one could optimize a classification algorithm for each class in a way that binary classification be performed separately as one-against-all approach.

VI. CONCLUSION

In this study, we proposed a DCGAN and Capsule Network-based framework (referred to as WaferCaps) to generate synthetic WBM images and classify them according to eight different defect patterns, namely, Center, Donut, Edge-Loc, Edge-Ring, Loc, Near-full, Scratch and Random. For this purpose, labelled and patterned dataset of WM-811k data were used. DCGAN was utilized first in order to upsample the data such that each class will be increased into 10,000 samples. A different method was used to upsample the Scratch class, in which the defect pattern was isolated and rotated into different angles to increase the number of defect scenarios.

Two main datasets were then created for our analysis namely, original and mixed. The original dataset contained the WBMs available in the WM-811k dataset only, while the mixed dataset contained synthetic and original WBMs together with maintaining the test set the same for both of them which contained original samples. This process was essential in order to observe the effect of using synthetic WBMs generated by DCGAN on the test accuracy.

The mixed dataset were then used on the CapsNet with different parameters and the proposed WaferCaps were obtained according to the best scenario of a specified series of high impact parameters that lead to the best test accuracy. The WaferCaps performance was then compared with the occasion when the original dataset was used for training. Our experiments showed that the proposed WaferCaps achieved a training, validation and test accuracies of 99.59%, 97.53% and 91.41% respectively when mixed dataset was used. While it achieved a training, validation and test accuracies of 99.89%, 88.92% and 78.2% respectively when only original dataset was used.

Also we compared the performance of our proposed WaferCaps with different deep learning models such as the original CapsNet proposed by Sabour *et al.* [24] to classify MNIST handwritten digits, CNN, and MLP. The experiments show that our proposed WaferCap outperformed all the other deep learning models that were compared to when mixed dataset was used.

In follow-up research, we will focus on how to optimize DCGAN to generate realistic scratch pattern defects and to optimize WaferCaps classification performance for each defect class separately.

References

[1] A. A. R. M. A. Ebayyeh and A. Mousavi, "A review and analysis of automatic optical inspection and quality monitoring methods in electronics industry," *IEEE Access*, vol. 8, pp. 183192–183271, 2020.

[2] J. Yu, "Enhanced stacked denoising autoencoder-based feature learning for recognition of wafer map defects," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 4, pp. 613–624, Nov. 2019.

[3] J. Yu and X. Lu, "Wafer map defect detection and recognition using joint local and nonlocal linear discriminant analysis," *IEEE Trans. Semicond. Manuf.*, vol. 29, no. 1, pp. 33–43, Feb. 2016.

[4] C. H. Jin, H. J. Na, M. Piao, G. Pok, and K. H. 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.

[5] M.-J. Wu, J.-S. 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.

[6] H. Kahng and S. B. Kim, "Self-supervised representation learning for wafer bin map defect pattern classification," *IEEE Trans. Semicond. Manuf.*, vol. 34, no. 1, pp. 74–86, Feb. 2021.

[7] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[8] G. Douzas, F. Bacao, and F. Last, "Improving imbalanced learning through a heuristic oversampling method based on k-means and smote," *Inf. Sci.*, vol. 465, pp. 1–20, Oct. 2018.

[9] J. Yu and J. Liu, "Two-dimensional principal component analysis-based convolutional autoencoder for wafer map defect detection," *IEEE Trans. Ind. Electron.*, vol. 68, no. 9, pp. 8789–8797, Sep. 2021.

[10] T. Yuan and W. Kuo, "A model-based clustering approach to the recognition of the spatial defect patterns produced during semiconductor fabrication," *IIE Trans.*, vol. 40, no. 2, pp. 93–101, 2007.

[11] T. Yuan and W. Kuo, "Spatial defect pattern recognition on semiconductor wafers using model-based clustering and Bayesian inference," *Eur. J. Oper. Res.*, vol. 190, no. 1, pp. 228–240, 2008.

[12] C.-H. Wang, "Separation of composite defect patterns on wafer bin map using support vector clustering," *Expert Syst. Appl.*, vol. 36, no. 2, pp. 2554–2561, 2009.

[13] S. Park, J. Jang, and C. Kim, "Discriminative feature learning and cluster-based defect label reconstruction for reducing uncertainty in wafer bin map labels," *J. Intell. Manuf.*, vol. 32, no. 1, pp. 251–263, 2020.

[14] C.-S. Liao, T.-J. Hsieh, Y.-S. Huang, and C.-F. Chien, "Similarity searching for defective wafer bin maps in semiconductor manufacturing," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 953–960, Jul. 2014.

[15] 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.

[16] 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.

[17] J. Yu, X. Zheng, and J. Liu, "Stacked convolutional sparse denoising auto-encoder for identification of defect patterns in semiconductor wafer map," *Comput. Ind.*, vol. 109, pp. 121–133, Aug. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016636151930106X

[18] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[19] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[20] S. K. Venu and S. Ravula, "Evaluation of deep convolutional generative adversarial networks for data augmentation of chest X-Ray images," *Future Internet*, vol. 13, no. 1, p. 8, 2020.

[21] Y. Zhou, Q. Zhang, and V. P. Singh, "An adaptive multilevel correlation analysis: A new algorithm and case study," *Hydrol. Sci. J.*, vol. 61, no. 15, pp. 2718–2728, 2016.

[22] A. Punjabi, J. Schmid, and A. Katsaggelos, "Examining the benefits of capsule neural networks," 2020, *arXiv:2001.10964*.

[23] F. Deng, S. Pu, X. Chen, Y. Shi, T. Yuan, and S. Pu, "Hyperspectral image classification with capsule network using limited training samples," *Sensors*, vol. 18, no. 9, p. 3153, 2018.

[24] S. Sabour, N. Frosst, and G. Hinton, "Dynamic routing between capsules," in *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc. Inc., Dec. 2017, pp. 3859–3869. [Online]. Available: https://dl.acm.org/doi/10.5555/3294996.3295142

[25] H. Wang, K. Shao, and X. Huo, "An improved CapsNet applied to recognition of 3D vertebral images," *Appl. Intell.*, vol. 50, no. 10, pp. 3276–3290, 2020.

[26] M. K. Patrick, A. F. Adekoya, A. A. Mighty, and B. Y. Edward, "Capsule networks—A survey," *J. King Saud Univ. Comput. Inf. Sci.*, to be published.

[27] Z. Zhu, G. Peng, Y. Chen, and H. Gao, "A convolutional neural network based on a capsule network with strong generalization for bearing fault diagnosis," *Neurocomputing*, vol. 323, pp. 62–75, Jan. 2019.

[28] M. Hashemi, "Enlarging smaller images before inputting into convolutional neural network: Zero-padding vs. interpolation," *J. Big Data*, vol. 6, no. 1, p. 98, 2019.

[29] S. Cai, J. Gao, M. Zhang, W. Wang, G. Chen, and B. C. Ooi, "Effective and efficient dropout for deep convolutional neural networks," 2019, *arXiv:1904.03392*.