Journal of Intelligent Manufacturing (2025) 36:3359–3397
https://doi.org/10.1007/s10845-024-02377-4

Sparse deep encoded features with enhanced sinogramic red deer optimization for fault inspection in wafer maps

Doaa A. Altantawy[1] · **Mohamed A. Yakout**[1]

Received: 21 September 2023 / Accepted: 17 March 2024 / Published online: 20 May 2024

Abstract

Due to the complexity and dynamics of the semiconductor manufacturing processes, wafer bin maps (WBM) present various defect patterns caused by various process faults. The defect type detection on wafer maps provides information about the process and equipment in which the defect occurred. Recently, automatic inspection has played a vital role in meeting the high-throughput demand, especially with deep convolutional neural networks (DCNN) which shows promising efficiency. At the same time, the need for a large amount of labeled and balanced datasets limits the performance of such approaches. In addition, complex DCNN in recognition tasks can provide redundant features that cause overfitting and reduce interpretability. In this paper, a new hybrid deep model for wafer map fault detection to get over these challenges is proposed. Firstly, a new convolutional autoencoder (CAE) is employed as a synthetization model to fix the high imbalance problem of the dataset. Secondly, for efficient dimensionality reduction, an embedding procedure is applied to the synthesized maps to get sparse encoded wafer maps by reinforcing a sparsity regularization in an encoder-decoder network to form a sparsity-boosted autoencoder (SBAE). The sparse embedding of wafer maps guarantees more discriminative features with 50% reduction in spatial size compared to the original wafer maps. Then, the 2D encoded sparse maps are converted to 1D sinograms to be fed later into another aggressive feature reduction stage using a new modified red deer algorithm with a new tinkering strategy. The resultant feature pool is reduced to ~ 25 1D feature bases, i.e., ~ 1.5% of the initial size of the 2D wafer maps. Finally, for the prediction stage, a simple 1DCNN model is introduced. The proposed inspection model is tested via different experiments on real-world wafer map dataset (WM-811K). Compared to state-of-the-art techniques, the proposed model outperforms their performance even with small-sized 1D feature pool. The average testing accuracy are 98.77% and 98.8% for 9 and 8 types of faults, respectively.

Keywords Wafer maps · WM-811K · Autoencoders · Sparse encoding · 1DCNN · Red deer optimization

Introduction

In semiconductor manufacturing, wafer is an important fundamental component in integrated circuits (IC). A single wafer can involve several hundred integrated circuits (ICs) after hundreds of sophisticated processes (Alam & Kehtarnavaz, 2022). Any abnormality in these production processes may lead to the generation of defects in the wafer map. Hence, due to the complexity of these processes, it is impossible to produce wafers without any defects (Jin et al., 2019; Liu & Chien, 2013).

After completing the wafer fabrication processes, every wafer undergoes a testing procedure including a series of multiple electrical tests to determine whether each individual chip (or die) meets its product specifications. Specifically, a probe test bench is utilized to detect the electrical characteristics of the chip dies (Cheng et al., 2021). Then, according to the quality level, the chip dies are marked in different colors on the wafer map. Typically, the captured defect patterns are divided into two types: global random defects and local systematic defects. In the global ones, defects are distributed randomly across the wafer without any spatial arrangement,

✉ Doaa A. Altantawy
doaa1adel@mans.edu.eg

Mohamed A. Yakout
myakout@mans.edu.eg

[1] Electronics and Communications Engineering Department, Faculty of Engineering, Mansoura University, 60 El-Gomhoria Street, Mansoura, Egypt

even in normal production conditions. In contrast, in the local ones, spatial correlations are observed in specific regions of a wafer, resulting in patterns such as center circles, edge rings, local zones, and scratches (Wu et al., 2014).

Integrated circuits manufacturing requires high investment, precise technology, and a complex manufacturing process. Thus, an analysis of the wafer map is essential to improve the yield, quality, and reliability of the IC manufacturing process. Even so, manually annotating wafer maps with their defect types is time-consuming and expensive, especially with large production lines (Shankar & Zhong, 2005). Moreover, engineers judge the defect types of wafer map based on their professional knowledge and work long hours which can be exposed to visual fatigue and raises the risk of erroneous classification. Hence, automatic inspection of the wafer map defect is a necessary step which can reduce the time and cost.

With the advancements of machine learning and deep learning algorithms, building an effective automatic fault detection model has become a hot topic in the research community (Kim & Behdinan, 2023; Theodosiou et al., 2023). These wafer fault detection models can be categorized into segmentation models (Cheng et al., 2021; Chu et al., 2022; Jin et al., 2019; Kim & Kang, 2021; Lee et al., 2010; Nag et al., 2022; Yan et al., 2023) and classification models (Baly & Hajj, 2012; Kyeong & Kim, 2018; Saqlain et al., 2019; Yu et al., 2019; Jin et al., 2020; Chen et al., 2021, Chen et al., 2022; Kang & Kang, 2021; Kim et al., 2021; Wang et al., 2021, Wang et al., 2022; Yu et al., 2021a, b, c; Zheng et al., 2021; Shin et al., 2022; Xuen et al., 2022; Yoon & Kang, 2022; Yu et al., 2022; Zhang et al., 2022; Alqudah et al., 2023). The classical supervised recognizers have achieved some good results in wafer map defect recognition (Alqudah et al., 2023; Baly & Hajj, 2012; Cheng et al., 2021; Saqlain et al., 2019). Nevertheless, their performances relied on the effectiveness of the feature extraction step. In addition, the spatial resolution and noise of the wafer maps significantly affect the performance of such techniques. Accordingly in recent times, deep feature learning and extraction has been widely applied in the field of wafer defect recognition (Kyeong & Kim, 2018; Yu et al., 2019, 2022; Jin et al., 2020; Chen et al., 2021, Chen et al., 2022;; Kang & Kang, 2021; Kim et al., 2021; Wang et al., 2021, Wang et al., 2022; Yu et al., 2021a, b, c; Zheng et al., 2021; Shin et al., 2022; Xuen et al., 2022; Yoon & Kang, 2022; Zhang et al., 2022; Xu et al., 2023). However, employing the traditional 2D convolutional neural network (CNN) in the direct classification of defects can easily lead to instability in the classification results (Xu et al., 2023), especially with very small image resolution, like the employed WM-811K wafer map dataset. This instability occurs because of the lack of spatial information to learn features and make proper predictions. In addition, 2DCNNs generally releases high dimensional features that may contain many redundant information, which brings some challenges at the final classification stage, like increasing the computational complexity, reducing memory efficiency, and increasing the chance of overfitting (Yu et al., 2019; Jin et al., 2020; Chen et al., 2021, Chen et al., 2022; Kang & Kang, 2021; Wang et al., 2021, Wang et al., 2022; Yu et al., 2021a, b; Zheng et al., 2021; Xuen et al., 2022; Yoon & Kang, 2022; Yu et al., 2022; Zhang et al., 2022). Many studies have tried different feature engineering steps with deep models to get over these challenges in (Jin et al., 2020; Yu et al., 2021b; Zheng et al., 2021). Table 1 summarizes the most recent studies that employ deep models, indicating the main procedure, strengths, and issues in each one.

Accordingly, the main target in this work is how to build a deep recognition system that can provide precise salient features, despite the very low-resolution of wafer maps, with the highest recognition performance. In addition, this recognition system can avoid redundant information that leads to instability and bad interpretability. Thus, we go for converting the 2D wafer map fault detection problem to 1D detection one to get the most salient features with the least dimensionality. Therefore, the redundancy and the high dimensionality of features can be reduced. Nonetheless, new challenges are raised, such as how to assign suitable embedding that kept the 2D spatial information in 1D representation, proper 1D feature ranking, and suitable 1DCNN classifier. Accordingly, the main contributions of the proposed wafer map defect classification model can be summarized as follows.

(1) We have exploited the importance of encoder-decoder networks, i.e., autoencoders (AE), in two ways. First, AE is employed as a new convolutional synthetization model to get over the high imbalance problem in the employed wafer map dataset. This model proves efficiency by reconstructing the original wafer maps with a total loss of 0.0011. Second, AE is introduced in a new structure as an embedding representation step with dimensionality reduction behavior, named as sparsity-boosted autoencoder (SBAE). The resultant encoded sparse maps from SBAE guarantee more discriminative features with a 50% reduction in size compared to the original wafer maps. Despite this reduction in size, inspection accuracy of 99.48% can be obtained while working with initial wafer map resolution of 27×25.
(2) An enhanced red deer optimization (ERD) with a new tinkering strategy is proposed. ERD is applied to 1D squeezed sinograms of the previous sparse maps. The ERD algorithm results in a final average feature pool of ~ 15 bases, i.e., ~ 1.5% of the initial wafer map size which has resolution of 33×29. The performance of ERD is compared, in an ablation behavior, to other different

Table 1 Summary of the state-of-the-art techniques in wafer map fault detection following hybrid methods in inspection

Method	Main procedure	Strengths	Issues
Jin et al. (2020)	They utilized convolutional neural network to extract high-level features and then features were fed to combination of error-correcting output codes (ECOC)and support vector machines (SVM)	They employed different traditional classifiers with CNN and ECOC in ablation study. They achieved overall classification accuracy of up to 98.43% via 10-fold cross validation	The computation effort is large to extract high-level features from CNN, besides adjusting the hyperparameters of CNN and ECOC is difficult
Yu et al. (2021b)	They extracted high-dimensional features using a pretrained DenseNet, and for the classification stage, they have assigned deep forest	They got over the redundancy and the high dimensionality of the extracted features through employing Multi-grained cascade forest (GCForest). They achieved achieves a correct recognition rate of 96.2% on the testing dataset	GCForest is a computationally expensive algorithm to train and use. It is highly dependent on the quality of the training data. If the training data is noisy or incomplete, GCForest may not perform well
Zheng et al. (2021)	They developed their own deep convolutional model	It is a simple deep learning model. They compared their model to other machine learning models. They gained a maximum accuracy of 93.75%	Building a CNN from scratch requires a significant amount of time and effort. There are many parameters and hyperparameters that need to be tuned carefully to achieve optimal performance
Chen et al. (2020)	They extracted multi-source features from two different dual-channel deep networks (DCNN). These features were fed into ECOC–SVM classification model	Applying two parallel channels, DCNNs can simultaneously extract both local and global features from the input data, which enhances the quality of features, and their robustness to noise. ECOC–SVM is a multi-class classification algorithm that combines the strengths of support vector machines (SVMs) and error-correcting output codes (ECOC). They declared accuracy of 96.4%	DCNNs are complex with a large number of parameters. ECOC–SVM can be more computationally complex than other multi-class classification algorithms, especially for large datasets with many classes. It is also sensitive to imbalanced datasets
Kang and Kang (2021)	They used CNN with ML models to extract handcrafted and convolutional features. These features are fed into multi-response linear regression (MLR) as meta-classifier for the final prediction stage	They can extract salient features through their stacking ensemble	They extracted manually handcrafted features to feed both the CNN and ML models MLR assumes that the relationship between the predictors and the outcomes is linear Outliers can distort the results of MLR, leading to inaccurate predictions
Chen et al. (2022)	They built a dual-source DCNN structure which is fed with the original wafer maps and the preprocessed ones. Then, an information entropy fusion was performed to the resultant dual-source probability to get the final prediction	The fusion procedure using information entropy from two sources help to extract more salient features. They declared accuracy of 98.34%	Having two networks means higher computational cost because of the number of trainable parameters
Yu et al. (2019)	They developed stacked convolutional sparse denoising auto-encoder (SCSDAE) to generate effective salient features from wafer maps. These features were fed into SVM classifier to predict the defect label	They produced salient features with high robustness to noise. They declared a maximum accuracy of 95.13%	Training SVMs can be computationally expensive, especially with large datasets with many features. The choice of kernel function significantly impacts the performance of the SVM. Selecting the appropriate kernel function can be challenging and requires experimentation

Table 1 (continued)

Method	Main procedure	Strengths	Issues
Wang et al. (2021)	They used a variational autoencoder (VAE) and decoders to generate similar wafer defect maps and a refined deep convolutional neural network (CNN) for feature learning	The VAE guarantees effective embedding of features to be fed later into a deep CNN model. They declared accuracy of 99.16% for the synthesized data versus 96.20% for the original one	VAEs often require a high-dimensional latent space to capture the complexity of the data. Training VAEs can be computationally expensive, especially for large datasets with high-dimensional latent spaces. This is because VAEs require optimizing both the encoder and decoder networks, as well as the variational loss function
Wang et al. (2022)	They assigned a convolutional autoencoder for balancing data, then, for the classification stage they employed a deep CNN network with residual block	Employing residual block helps to increase the model depth, hence effective feature representations can be obtained. They achieved an accuracy of 99.9%	Residual blocks increase the model size by adding more additional parameters compared to traditional convolutional blocks. In addition, residual blocks can make networks more prone to overfitting
Yu et al. (2021a)	They developed a deep transfer Wasserstein adversarial network (DTWAN) with multi-stage optimization	They have shown superiority to typical transfer learning algorithms in wafer inspection through employing generative adversarial algorithm to guide the model to extract general features from source and target domain. They declared recognition rate from 0.7 to 0.98	Training WANs is more computationally expensive than traditional networks. They are very sensitive to noise and the choice of hyperparameters
Zhang et al. (2022)	They utilized a binarized neural network (BNN) to reduce runtime and memory consumption while achieving high accuracy in the recognition of wafer defect	Compared to the CNN model with the same architecture, Their BNN model achieved 1.66 × speedup and 29.70 × memory reduction. They declared accuracy of 94.63%	Replacing real-valued weights and activations with binary values in BNN inevitably leads to a loss of precision, which can negatively impact the accuracy of the network
Yoon and Kang (2022)	They developed a semi-automatic procedure for the defect recognition by using uncertainty quantification. Hence, it was decided whether to use manual recognition or CNN-based recognition for the inserted wafer map	It is the first work to adopt an uncertainty-based reject option to selectively utilize CNN. Instead of using CNN only, the proposed method obtained assistance from a process engineer to fulfill the near-perfect accuracy requirement. They declared accuracy of 96.73% with 100% coverage of CNN	Uncertainty quantification can introduce additional computational overhead compared to traditional CNNs. In addition, it may lead to a slight decrease in prediction accuracy compared to unquantified CNNs

metaheuristic algorithms, such as Genetic (GA), Equilibrium (EO), Grey Wolf (GWO), Sine cosine (SCA), and particle swarm algorithms (PSO). The proposed ERD achieves the least feature pool size with approximately the same accuracy as its alternatives, because of the proposed tinkering strategy that makes the ERD algorithm reaches the global optimum solution with the least number of discriminative features to avoid any possible redundant information.

(3) Intensive experiments, with a new predictive 1DCNN model, are performed on different resolutions of wafer maps in 8- and 9-fault type prediction. An average accuracy of 95.2% is achieved for unseen 62% testing part of the dataset, while an average accuracy of 98.1% is achieved for unseen 20% testing part of the dataset of in a train–test–validation evaluation. Despite the aggressive dimensionality reduction, the proposed inspection model proves efficient generalization. In addition, the proposed 1DCNN network proves a great balance between the number of parameters and the targeted accuracy in fault detection compared to other common 1DCNNs, such as 1D-VGG16, 1D-ResNet50, 1D-LeNet-5, and 1D-Inception.

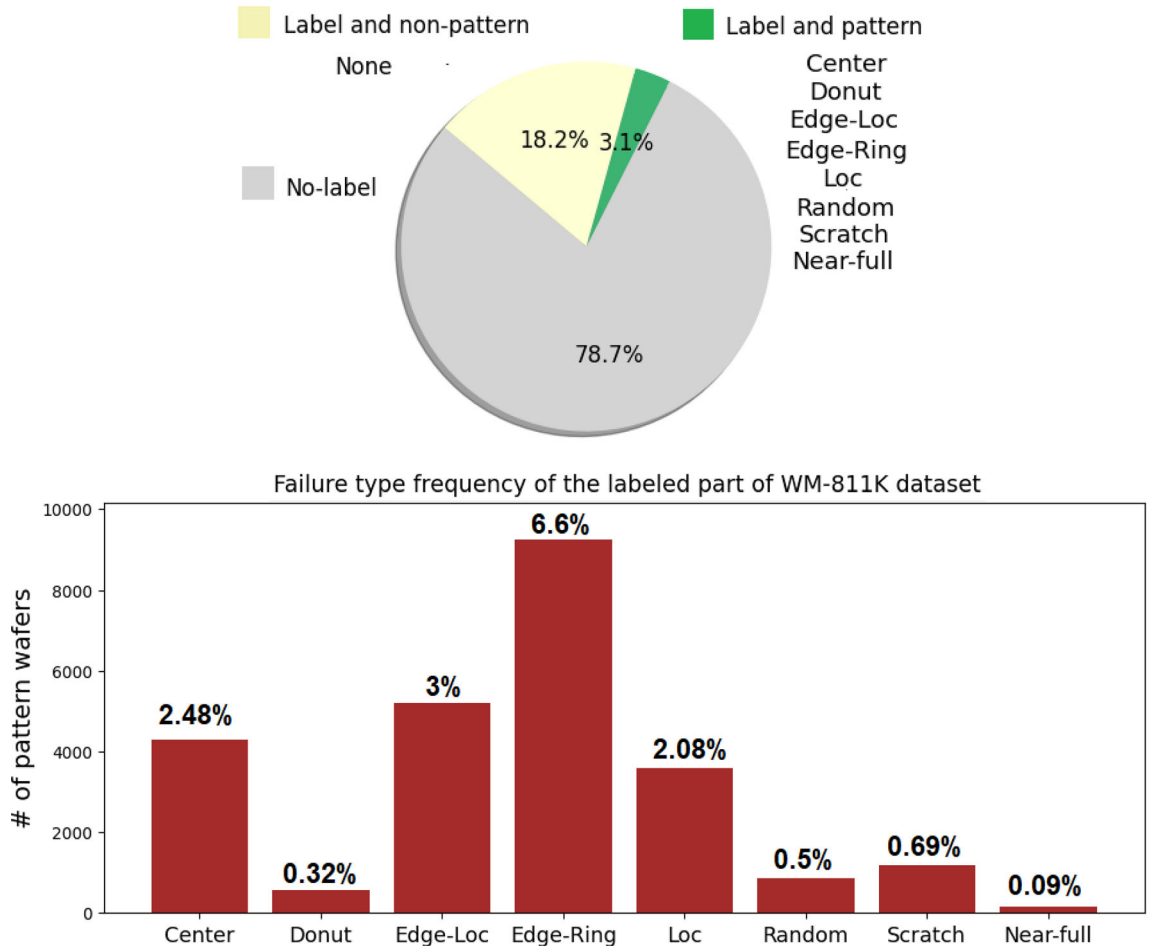


Fig. 1 Labeling and categories distribution of WM-811K dataset

The rest of the paper is organized as follows. Details about the targeted wafer map dataset are demonstrated in "Wafer map dataset" section. "Methodology" section describes the details of the proposed methodology of wafer map fault detection. "Experimental results and discussion" section indicated the performed experiments with its results. Finally, the conclusion is offered in "Conclusion" section.

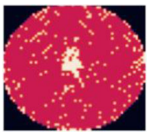
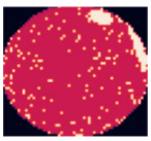
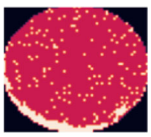
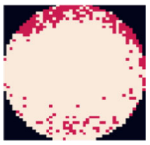
Wafer map dataset

The WM-811K (Wu et al., 2014) is the employed wafer map dataset and it is publicly available at Kaggle website (WM-811K, 2014) It contains 811,457 instances collected from 46,293 lots during the semiconductor fabrication process (Wu et al., 2014). Only a subset of 21.3% (172,950) is labeled by professionals with one of the following nine categories: Center, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, Near-Full, and None, while the rest of the group is still unlabeled, check Fig. 1. As indicated from Fig. 1, the dataset is mostly unlabeled and most of the labeled part is free of fault "None" and the faulty part is very imbalanced. In Table 2, the distribution and the main cause of different defects are pointed out. This dataset provide a single-type defect in a single wafer map (Baly & Hajj, 2012; Saqlain et al., 2019; Yu et al., 2019, 2022; Jin et al., 2020; Chen et al., 2021; Kang & Kang, 2021; Wang et al., 2021, Wang et al., 2022; Yu et al., 2021a, b; Zheng et al., 2021; Chen et al., 2022; Xuen et al., 2022; Yoon & Kang, 2022; Zhang et al., 2022; Alqudah et al., 2023), but there are multiple studies that target mixed-type defect patterns in their inspection models, such as the ones in Kyeong et al. (2018), Kim et al. (2021), Sin et al. (2022), Yu et al. (2022), and Xu et al. (2023).

Methodology

The main steps of the proposed model for wafer map fault detection are shown in the graphical abstract of Fig. 2, which combines a flow chart with a design purpose for each block.

Table 2 The observed defect patterns in WM-811K with counts and causes

Defect pattern	Wafer map example	Count (Proportion with respect to the labelled group)	The cause of the defect
Center		4294 (2.48%)	It is caused by abnormal deposition, fluid flow, or pressure process
Donut		555 (0.32%)	It is caused by the precipitation of the dissolved photoresist solids on the wafer surface
Edge-loc		5189 (3%)	It is caused by abnormal film deposition
Edge-Ring		9680 (5.6%)	It is caused by abnormal temperature during annealing
Loc		3593 (2.08%)	It is local error caused by excessive machine vibration
Random		866 (0.5%)	It is caused by airborne dust interference
Scratch		1193 (0.69%)	It is a mechanical damage happening during handling or cutting
Near full		149 (0.09%)	It is caused by Human error
None		147,431 (85.25%)	It is a normal wafer map with some noise

These steps are detailed in the following subsections. Algorithm 1 summarizes a pseudo code for the whole proposed wafer fault detection model.

Wafer data synthetization model

As presented in Table 2 and Fig. 1, WM-811K is a highly imbalanced dataset. Each wafer map consists only of three types of pixels: 0 for the background, 1 for the normal pixels and 2 for the defected ones. Consequently, two main preprocessing steps are performed. The first is one-hot encoding to convert the grey wafer maps $X(m, n)$ to colored ones $XX(m, n, c)$, where c is the number of channels, as $xx\left(m, n, c_{i=x(m,n)}\right) = 1, where\ i \in \{0, 1, 2\}$, $x(m, n)$ and $xx(m, n, c)$ denotes the grey and colored pixel value. RGB wafer maps help to extract multi-scale features in the upcoming procedures. The second preprocessing step is a synthetization (augmentation or balancing) model. In this work, an autoencoder-based synthesizing model is used for data augmentation.

An autoencoder (AE; Li et al., 2023) is a type of ANN that learns efficient mappings or codings of unlabeled data in an unsupervised manner. AE extracts output data to reconstruct input data and compare it with original input data. After numerous times of iterations, the value of cost function reaches its optimality, which means that the reconstructed input data is able to approximate the original input data to a maximum extent. The introduced convolutional autoencoder (CAE) shows superiority to the traditional AE by incorporating convolutional layers which preserves the local image structures by incorporating spatial relationships between pixels in images.

The introduced CAE consists mainly of two parts: encoder and decoder, see Fig. 3. The encoder converts the input map to a bottleneck low dimensional feature map, while the decoder performs deconvolution operations to expand the latent feature map to reconstruct the original wafer map. The encoder in the proposed architecture consists of one 2D convolutional layer with 64 filters with a kernel weight of (3×3), and a

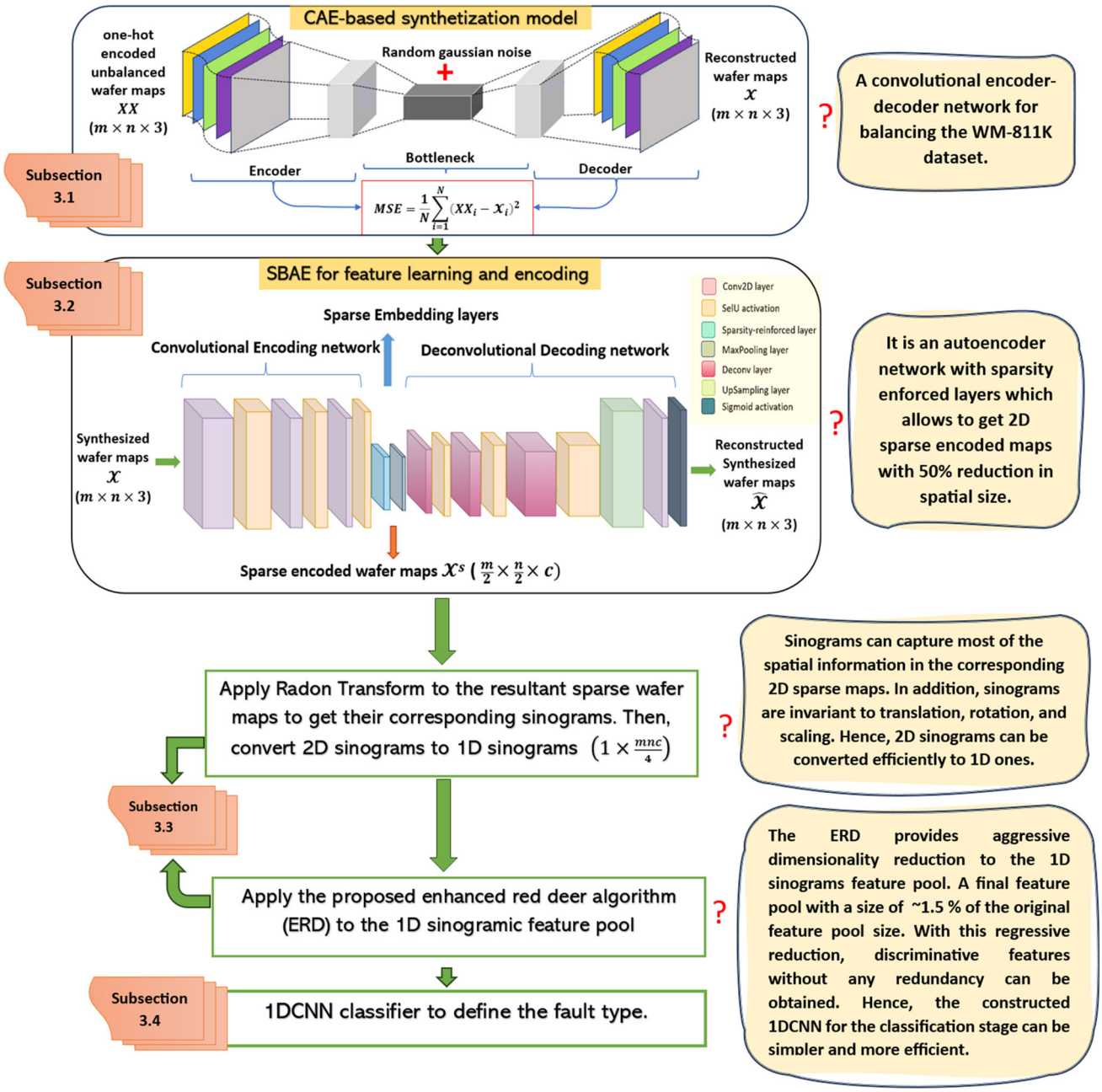


Fig. 2 Graphical abstract of the proposed fault type prediction in wafer maps

MaxPooling layer. The extracted feature maps from the convolutional layer of the encoder are represented as

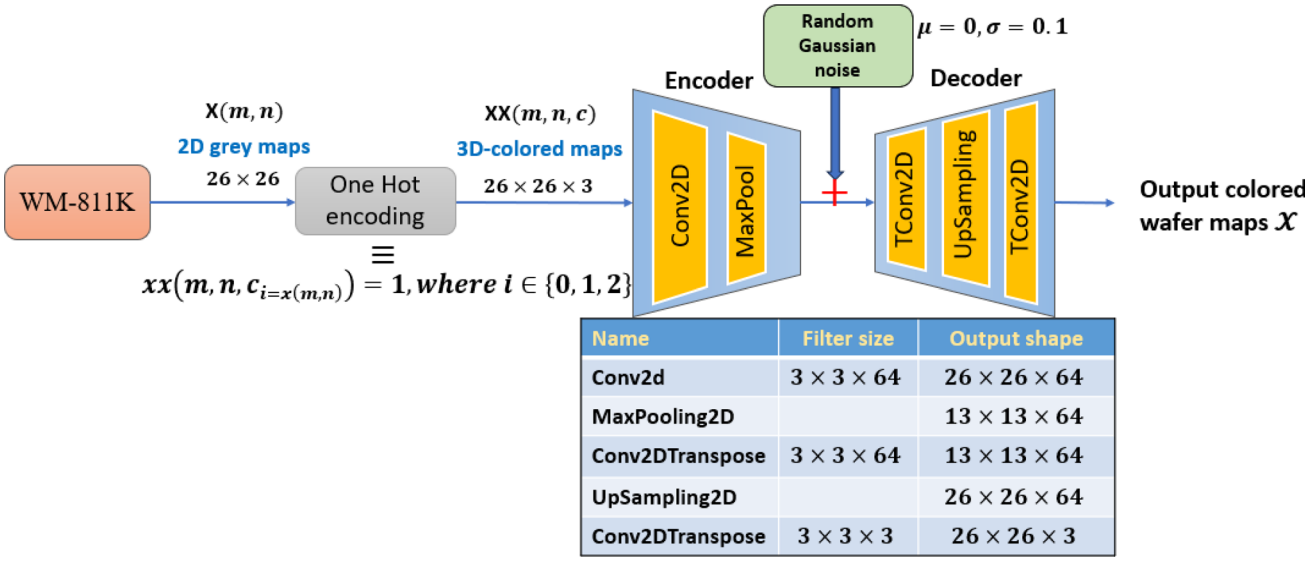
$$H = \mathcal{A}(XX * W + B), \tag{1}$$

where $\mathcal{A}$ is the activation function which is employed as ReLU. XX is the encoded colored wafer maps. W and B are the weights (convolutional kernel) and the bias, respectively.

The extracted feature maps from the convolutional layer is fed into a MaxPooling layer to provide the targeted bottleneck low dimensional feature map H_c, as

$$H_c = \max_{i=0,\ldots,\nabla-1,\, j=0,\ldots,\nabla-1} H\left(x'+i,\, y'+j\right), \tag{2}$$

where ∇ is the MaxPooling operator. x', $and y'$ are the pixels coordinates. At this point, random Gaussian noise ($\mu = 0$, $\sigma = 0.1$) is added at the bottleneck embedded map H_c to provide more robustness to the synthetization model. Now, the decoder network tries to construct the input maps XX



Name	Filter size	Output shape
Conv2d	3 × 3 × 64	26 × 26 × 64
MaxPooling2D		13 × 13 × 64
Conv2DTranspose	3 × 3 × 64	13 × 13 × 64
UpSampling2D		26 × 26 × 64
Conv2DTranspose	3 × 3 × 3	26 × 26 × 3

Fig. 3 The CAE-based synthetization model for wafer maps augmentation

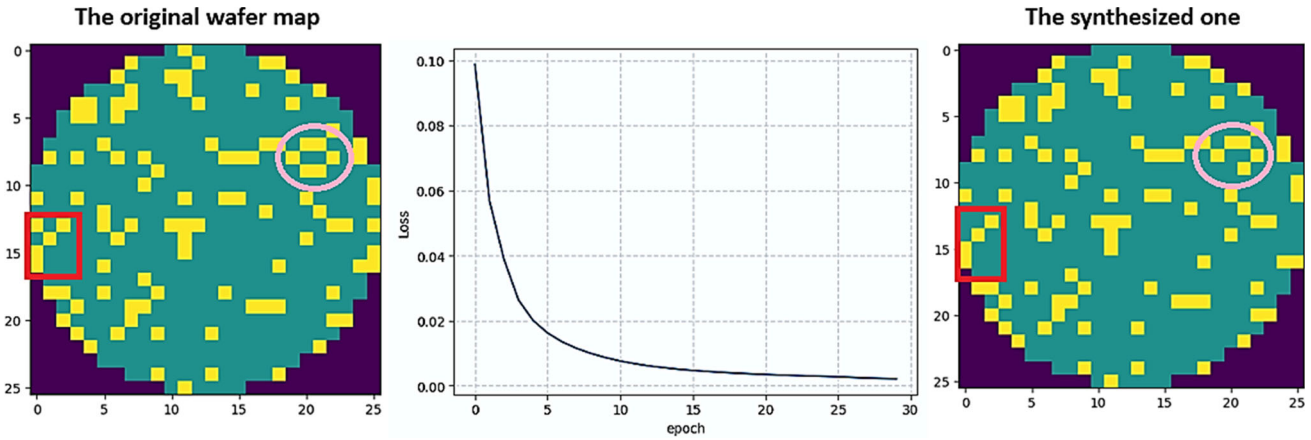


Fig. 4 The performance of the proposed CAE-based synthetization model—Sample wafer map with its synthesized one and the training reconstruction error over epochs

through employing two transposing convolution (deconvolutional) layers and UpSampling layer, revise Fig. 3. The output of decoder is the restored feature maps of H_c as

$$\mathcal{X} = \mathcal{A}(H_c * / * W + B), \tag{3}$$

where $*/*$ is the deconvolution (Conv2DTranspose) operator. The first Conv2DTranspose (TConv2D) layer in the decoder network employs 64 filters with a kernel weight of (3×3) and activation function $\mathcal{A}$ as ReLU. The second one employs three filters with a kernel weight of (3×3) and activation function $\mathcal{A}$ as Sigmoid. The proposed CAE is trained over certain epochs to minimize the reconstruction error between the original wafer maps and the synthesized ones, in terms of the mean squared error as

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(XX_i - \mathcal{X}_i)^2, \tag{4}$$

where $\mathcal{X} == \widehat{XX}$ is the new synthesized wafer maps and N is the number of the inserted wafer maps. Figure 4 indicates a sample wafer map and its corresponding synthesized one. As indicated, they look very similar in the indicated zoomed-up versions of maps. In addition, in the same figure, the reconstruction error over the training epochs is shown with a final loss 0.0011at epoch 30.

Sparse feature learning and encoding

At this stage, as low-dimensional 2D wafer maps are targeted, a new sparse autoencoder model is introduced. Sparsity is the property of being sparse or having a lot of zero entries (Sun et al., 2022). In the context of machine learning, sparsity is often used to refer to the number of zero weights in a neural network. An autoencoder is called sparse when its hidden layer activations are encouraged to be sparse (Ng, 2011). The sparsity constraint is added to the loss function of the traditional convolutional autoencoder. The sparsity constraint can be based on the L_1 norm of the hidden layer activations or on the KL divergence between the distribution of activations in the hidden layer and a target distribution that is sparse. For the main difference between the traditional autoencoder and the sparse one, check Fig. 5.

In the proposed sparsity-boosted autoencoder (SBAE), a sparsity-reinforced layer is added to the last layer of the encoding phase. Therefore, the network is encouraged to learn an encoding by activating only a small number of nodes. The cost function of the proposed SBAE utilizes three terms: a reconstruction term combined with other two regularizers, i.e., a weight decay term and another sparsity-boosting term. The reconstruction term is similar to the previous CAE. The weight decay term helps to decrease the magnitude of the weights and prevent overfitting. The sparsity-boosting term induces a sparsity penalty in the training criterion. For the configuration of the proposed SBAE, check Table 3.

Assume the synthesized wafer maps of N samples $(\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_N)$, where x_i represents the *ith* input of sample $\mathcal{X}^i$. The cost function considering only the reconstruction term with the weight decay term can be expressed as

Algorithm 1. The proposed wafer map fault detection

Input: $X(m,n)$ the original 2D grey wafer maps, learning rate = 0.0001, patch size = 32
Initialize colored wafer maps $XX(m,n,c) = 0$
Perform one hot encoding $xx\left(m,n,c_{i=x(m,n)}\right) = 1,\ where\ i \in \{0,1,2\}$
Construct the CAE in Fig. 3
Convolutional autoencoder (CAE) (XX) // function for data balancing
    While epochs < 30 || reconstruction error (Eqn. 4) < 0.01
        Perform the convolutional encoding $XX \rightarrow H_c$ // Eqn. 1, 2
        Perform the convolutional decoding $H_c \rightarrow \widehat{XX}$ // Eqn. 3
    End
End CAE
Use the trained CAE model to synthesize maps for the unbalanced classes to get new synthesized wafer maps $\mathcal{X}$
Construct SBAE network following the configuration in Table 3
Sparsity-boosted autoencoder (SBAE) $(\mathcal{X})$ // function for dimensionality reduction by Sparse feature learning and encoding
    While epochs < 120 || reconstruction cost (Eqn. 6) < 0.02
        Perform the convolutional encoding $\mathcal{X} \rightarrow \mathcal{X}^s$ // like Eqn. 1, 2
        Perform the convolutional decoding $\mathcal{X}^s \rightarrow \widehat{\mathcal{X}}$ // like Eqn. 3
    End
End SBAE
Apply Radon transform to each 2D sparse encoded map $\mathcal{X}^s$ to get 1D sinogramic map y^s
Apply Algorithm 2 for performing a final aggressive feature reduction using the enhanced red deer optimization (ERD)
Construct the proposed 1DCNN classifier in Fig. 8
Perform a train-valid training using Adam.
Test the model with the unseen part
Output: The fault type per each wafer map

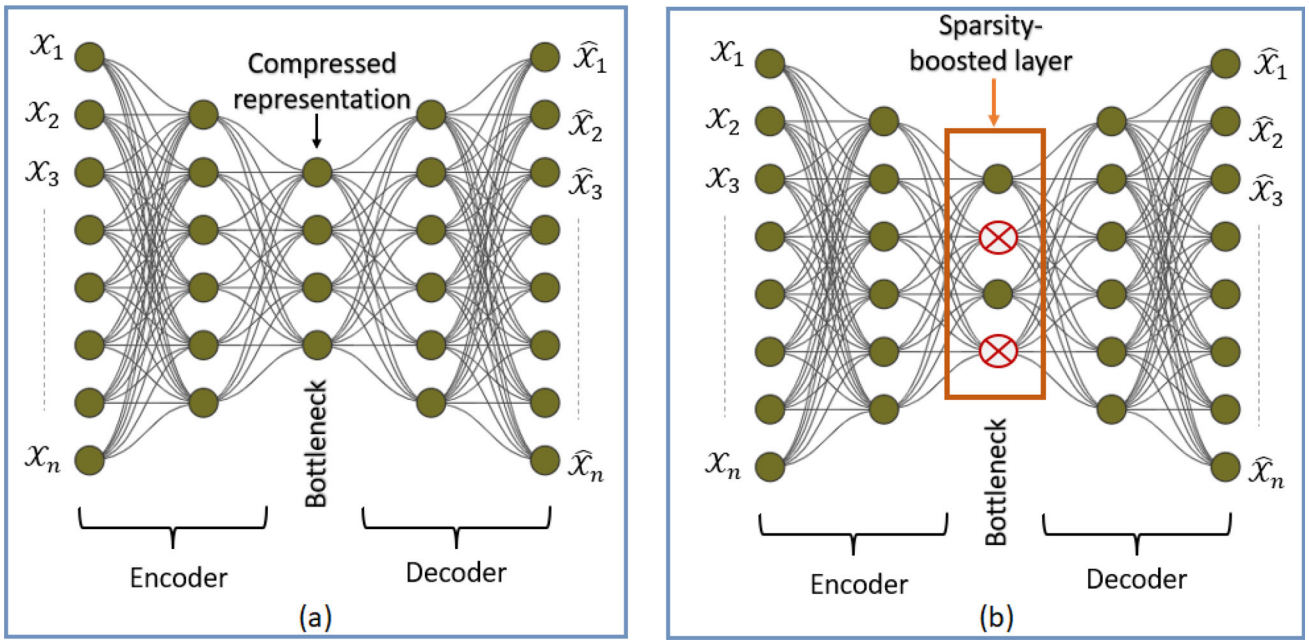


Fig. 5 Visual comparison between the traditional autoencoder in **a** and the sparsity-induced autoencoder in **b**. The dark green nodes are firing whereas the red nodes are constrained, i.e., we are in effect reducing the number of firing neurons

Table 3 Detailed configuration of the proposed SBAE

Task	Layer	Filter size	Output
Encoder network	Conv2d_1 + SeLU activation	$3 \times 3 \times 12$	$m \times n \times 12$
	Conv2d_2 + SeLU activation	$3 \times 3 \times 6$	$m \times n \times 6$
	Conv2d_3 + SeLU activation	$3 \times 3 \times 3$	$m \times n \times 3$
Bottleneck	Sparsity regularization	The sparsity parameter $\mathcal{P}$, the weight decay Ƕ, and the sparse penalty term γ were chosen to be 0.05, 0.0001, and 1	
	MaxPooling2D	(2×2)	$\frac{m}{2} \times \frac{n}{2} \times 3$
Decoder network	Conv2DTranspose_1 + SeLU activation	$3 \times 3 \times 3$	$\frac{m}{2} \times \frac{n}{2} \times 3$
	Conv2DTranspose_2 + SeLU activation	$3 \times 3 \times 6$	$\frac{m}{2} \times \frac{n}{2} \times 6$
	Conv2DTranspose_3 + SeLU activation	$3 \times 3 \times 12$	$\frac{m}{2} \times \frac{n}{2} \times 12$
	UpSampling2D	2×2	$m \times n \times 12$
	Conv2D_4 (Output) + Sigmoid activation	$4 \times 4 \times 3$	$m \times n \times 3$

$$\mathcal{C}(W,B) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} 0.5 \left\| \mathcal{A}_{W,B}\left(\mathcal{X}^i\right) - \mathcal{X}^i \right\|}_{reconstruction\ loss} + \underbrace{\frac{Ƕ}{2} \sum_{l=1}^{n_l-1} \sum_{i=1}^{o_l} \sum_{j=1}^{o_{l+1}} W_{ji}^{l}}_{weight\ decay\ term} \tag{5}$$

where $\mathcal{A}_{W,B}\left(\mathcal{X}^i\right) = \mathcal{A}\left(W * \mathcal{X}^i + \mathrm{B}\right)$ is the activation or mapping of the input $\mathcal{X}^i$ at layer l. n_l denotes the number of layers l in the targeted network. o_l is the number of nodes or units in layer l. Ƕ adjusts the weight of the decaying term; large Ƕ can cause overfitting, while small values may cause underfitting. In the proposed SBAE configuration, the value of Ƕadjusted empirically via multiple experiments.

For the sparsity-boosted term, the following term $\mathcal{K}\left(\mathcal{P} \| \widehat{\mathcal{P}}\right)$ is inserted in the cost function in Eq. 5 to be reformulated as

$$\mathcal{C}(W,B) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} 0.5 \left\| \mathcal{A}_{W,B}\left(\mathcal{X}^i\right) - \mathcal{X}^i \right\|}_{reconstruction\ loss} + \underbrace{\frac{\varsigma}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{o_l}\sum_{j=1}^{o_{l+1}} W_{ji}^{l}}_{weight\ decay\ term} + \underbrace{\gamma \sum_{j=1}^{o_l} \mathcal{K}\left(\mathcal{P}\middle\|\widehat{\mathcal{P}}_j\right)}_{sparsity\ term} \tag{6}$$

$$\mathcal{K}\left(\mathcal{P}\|\widehat{\mathcal{P}}_j\right) = \mathcal{P}\log\frac{\mathcal{P}}{\widehat{\mathcal{P}}_j} + (1-\mathcal{P})\log\frac{1-\mathcal{P}}{1-\widehat{\mathcal{P}}_j}, \tag{7}$$

where $\mathcal{K}\left(\mathcal{P}\|\widehat{\mathcal{P}}\right)$ is Kullback–Leibler divergence which seeks to reduce the deviation between $\widehat{\mathcal{P}}$ and $\mathcal{P}$. $\mathcal{P}$ denotes the targeted sparsity parameter, typically a small value close to zero. $\widehat{\mathcal{P}}$ is the average output of all hidden neurons, $\widehat{\mathcal{P}}_j = 1/N\sum_i \mathcal{A}_{W,B}\left(\mathcal{X}^i\right)$. γ is the sparse penalty coefficient.

Using the introduced SBAE, the wafer map of size $(m \times n \times c)$ is converted to a sparse encoded wafer map $\mathcal{X}^s$ of size $\frac{m}{2} \times \frac{n}{2} \times c$, which means the encoded map spatial size is reduced to the half. The encoded map $\mathcal{X}^s$ is obtained from the bottleneck layer of SBAE. To visualize the clustering performance of the resultant encoded maps, T-SNE (Van der Maaten & Hinton, 2008) is used, see Fig. 6. It helps to visualize the high-dimensional data by mapping the clustered features into low-dimensional space. As indicated, the encoded feature maps show better clustering performance compared to the original maps.

A new sinogramic red deer feature ranking

The main target of feature engineering steps is to reduce the feature dimensionality while keeping the best performance. At this stage, we intend to convert the sparse encoded wafer maps $\mathcal{X}^s$ to 1D signal without losing the spatial information of the 2D maps. Wherefore, the sparse encoded feature maps $\mathcal{X}^s$ are converted to sinograms by employing Radon transformation (Leavers, 1992). After that, each sinogram now can be converted to 1D signal y^s of size $\left(1 \times \frac{mnc}{4}\right)$, so for N samples, we have 1D feature pool Y^s of size $\left(N \times \frac{mnc}{4}\right)$. Then, the proposed enhanced red deer (ERD) algorithm is applied to the resultant sinograms to assign the optimal reduced features.

The conventional red deer algorithm

Red deer (RD) algorithm is a new nature-inspired optimization technique (Fathollahi-Fard et al., 2020). It belongs to the family of population-based metaheuristics algorithms. The main advantage of RD algorithm is that it equally maintains

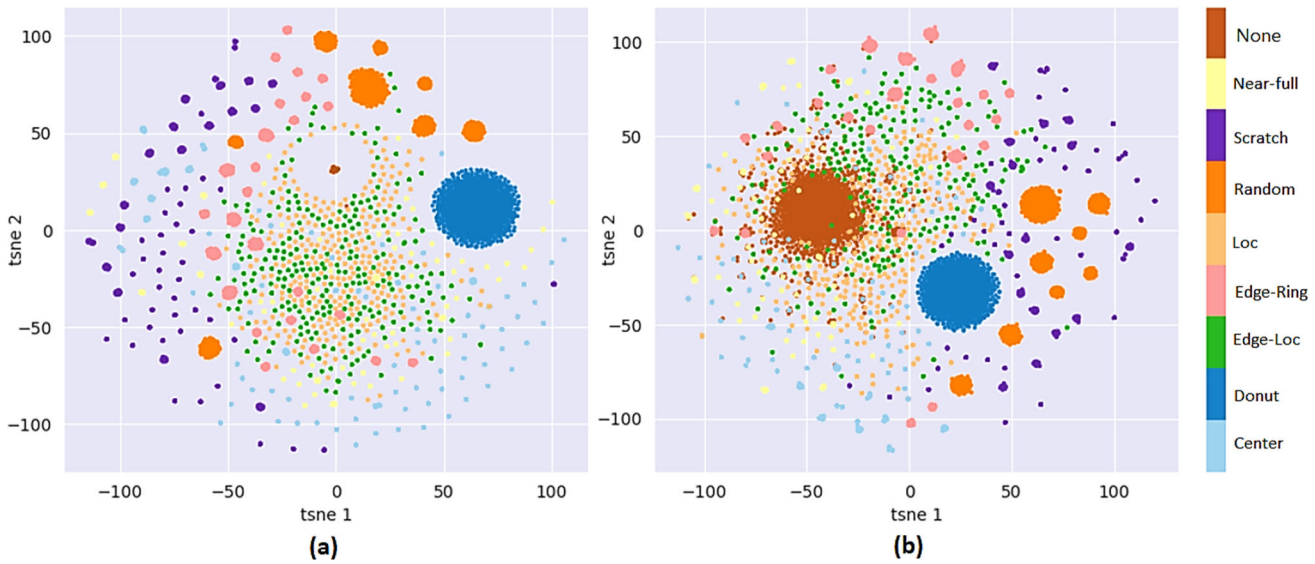


Fig. 6 t-SNE comparison before (**a**) and after (**b**) the sparse encoding by SBAE

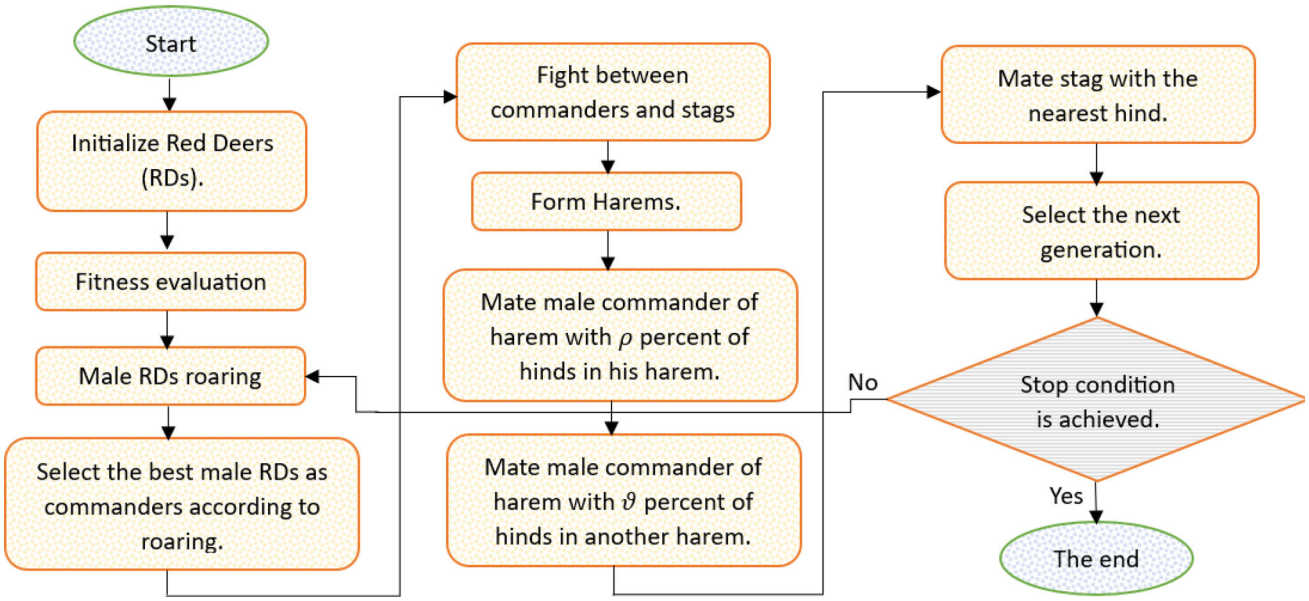


Fig. 7 Flow chart of the main Red Deer algorithm for sinogramic feature selection

the exploitation and exploration phases, which helps to assign the salient features with low complexity.

Red deer is male or female (hinds). A group of hinds is called a harem. Each harem is assigned a male commander. A competition is set among male RDs to get the harem with more hinds via roaring and fighting. According to the strength of the roaring phase, male RDs are categorized into commanders and stags. Only the strongest male after a fierce fight with the other males will be the commander of the harem. Here, the $\left(\frac{mnc}{4}\right)^{th}$ sparse encoded 1D features in $Y^S = \left[\mathcal{Y}_1,\ \mathcal{Y}_2, \ldots, \mathcal{Y}_{\frac{mnc}{4}}\right]$, where $\mathcal{Y}_i$ of size $(N,\ 1)$ is the feature vector, are considered as a group of RDs. Figure 7 indicates a flow chart of the RD algorithm. The main objective of optimization problem concentrates on the determination of near global or optimal solution evaluated with respect to the variables associated with the problem.

Stage 1: Initialize the population At this stage, an $\mathbb{N}_r$sparse sinogramic features to initialize red deers as, $G = \left[\mathcal{Y}_1,\ \mathcal{Y}_2, \ldots, \mathcal{Y}_{\mathbb{N}_p}\right]$,. Among this population, $\mathbb{N}_{male}$ are chosen as the male red deer features, G_{male}, while the rest $\mathbb{N}_{hind} = \mathbb{N}_p - \mathbb{N}_{male}$ are the hind group, G_{hind}. The best features in the selected population are selected as males according to their fitness value. The proposed objective or fitness function is a collaboration between the classification accuracy, based on KNN classifier, and the proportion of the selected number of red deers through a weighted sum as

$$f = \omega.acc + (1-\omega)\frac{\vartheta}{\Gamma}, \tag{8}$$

where acc denotes the classification accuracy of the currently selected red deers or features. ϑ represents the number of the currently selected red deers, while Γ is the total number of features in the targeted feature pool. ω is a weighting coefficient in the range of [0, 1].

Stage 2: Roaring phase The male agents are currently the superior solutions. Roaring is a local search for other neighboring best features. The updating rule is as follows.

$$RD_{male}^{new} = \begin{cases} RD_{male}^{old} + \alpha_1 \times ((\sqcap - \ell) * \alpha_2 + \ell),\ if \alpha_3 \geq 0.5, \\ RD_{male}^{old} - \alpha_1 \times ((\sqcap - \ell) * \alpha_2 + \ell),\ if \alpha_3 < 0.5, \end{cases} \tag{9}$$

where RD_{male}^{new} and RD_{male}^{old} are the current and the previous positions of male red deer solutions. $\sqcap$ and ℓ are the upper and lower limits of local search of neighboring solutions. α_1, α_2, and α_3 are randomly generated coefficients from a uniform distribution that ranges from 0 to 1.

Then, the male red deer solutions are categorized into commander and stage red deer solutions. The

number of male commanders, N_{com}, is considered as $\mathrm{N}_{com} = round(\delta \mathrm{N}_{male})$,, where δ is a random value in a range from 0 to 1. The number of stags can be expressed as $\mathrm{N}_{stag} = \mathrm{N}_{male} - \mathrm{N}_{com}$

Stage 3: The fighting phase between stags and commanders Here, every commander is allowed to fight with random stags. According to the solution space, we let the group solution of commanders G_{com} approaches that of stags G_{stag}. Accordingly, two new group of solutions are generated as

$$G_{new_1} = \frac{G_{com} + G_{stag}}{2} + \beta_1 \times ((u - \ell) * \beta_2 + \ell), \quad (10)$$

$$G_{new_2} = \frac{G_{com} + G_{stag}}{2} - \beta_1 \times ((u - \ell) * \beta_2 + \ell), \quad (11)$$

where G_{new_1} and G_{new_2} denote the new generated solutions due to the fighting process. u and ℓ are the limits of the search space. β_1 and β_2 are randomly generated coefficients from a uniform distribution that ranges from 0 to 1. Now, we have four solutions, i.e., G_{com}, G_{stag}, G_{new_1}, and G_{new_2}, the solution with the best cost function F will be selected as the final commander.

Stage 4: Forming harems Here, the assigned new commander is responsible for forming harems. A harem consists of a male commander and a group of female deer (hinds). The hinds are distributed into separate harems in a random manner, based on the power of the commander in roaring and fighting, i.e., it's fitting value. Hence, the number of hinds in a harem i, $\mathrm{N}_{hind}^{harem^i}$,, is calculated as

$$\mathrm{N}_{hind}^{harem^i} = round(f_i . \mathrm{N}_{hind}) \quad (12)$$

where f_i denotes the normalized power, fitting value, of the commander.

Stage 5: Mating phase After forming harems, there are three possibilities of mating. The first is that the commander of harem i mates with ρ of its harem's hinds, and the second occurs when the commander mates with ϑ of hinds of other harems. Commanders attack another harem to expand his command area. The third possibility is that each stag mates with the closest hind, regardless of harem restrictions. Due to the mating phase, new offspring RDs, G_{OS}, i.e., solutions, are generated as

$$G_{\mathrm{OS}} = \frac{G_{com} + G_{hind}}{2} + \theta \times (u - \ell), \quad (13)$$

where G_{com}, and G_{hind} are the group of solutions that represent commanders and hinds. θ is a random number between 0 and 1. In the third possibility of mating G_{com} is replaced by the stag solutions G_{stag}.

Algorithm 2. Detailed steps about the proposed enhanced red deer algorithm (ERD)

Initialize the RDs population, $\aleph_p = 50$

Input: $G = [\mathcal{y}_1, \mathcal{y}_2, \ldots, \mathcal{y}_{\aleph_p}]$ initial grouped features as red deers

Calculate the fitness of each red deer and sort them from the best to the worst according to f (Eqn. 5) based on KNN classifier
Select the best $\aleph_{male}$ as male red deers and the rest as hinds $\aleph_{hind}$ // $\aleph_{male} = 7,\ \aleph_{hind} = 43$
While $\ddot{\imath} < maxiter$ // $maxiter = 100$
  For each male RD
    Start the roaring phase, Eqn. 9
    Update the male RD position if it is better than previous considering the fitting value
  End for
  Categorize the males to stags and commanders $\aleph_{com} = round(\delta \aleph_{male})$, where $\delta = 0.6$, $\aleph_{stag} = \aleph_{male} - \aleph_{com}$
  For each commander
    Start the fighting between the commander and stags, Eqn. 10,11
    Update the positions
  End for
  Form harems, Eqn.12
  Assign the worst $\aleph_{harem}^{worst}$ harems, Eqn.14
  Define the targeted $\aleph_{harem}^{best}$ best harems, $\aleph_{harem}^{best} = \aleph_{harem}^{worst} + 3$
  Reformulate harems by using the tinkering strategy
  Compute the diversity in fitting v between the tinkered harem and the tinkering harem, Eqn.16
  For each commander
    Define ρ of the harem's hinds, and ξ of the new tinkering hinds // $\rho = 0.6, \xi = 0.5$
    If $v > \zeta$ // calculate v using Eqn. 16
      Start the mate phase with the original harem hinds to generate new offspring RDs/ or
      $$G_{\mathrm{OS}} = \frac{G_{com} + G_{hind}}{2} + \theta \times (u - \ell) \quad // u = 4, \ell = -4 \quad \text{(Eqn.15)}$$
    Elseif
      Start the mate phase with both original and tinkering hinds to generate new offspring RDs.
      $$G_{\mathrm{OS}} = \frac{G_{com} + G_{hind} + G_{tink}}{3} + \theta \times (u - \ell) \quad \text{(Eqn.15)}$$
    End if
  End for
  Define the nearest hinds to each stag
  For each stag
    Start the mate phase with its nearest hinds to generate new offspring RDs
    $$G_{\mathrm{OS}} = \frac{G_{stag} + G_{hind}}{2} + \theta \times (u - \ell)$$
  End for
  Define the next generation using the roulette wheel selection
  Update the feature pool if there is a better solution
  $\ddot{\imath} = \ddot{\imath} + 1$
End while
Output: optimized feature pool G^*

Finally, the next generation is assigned from all the best commanders (a certain percentage of the best solutions), and the best hinds from all hinds and offspring, via the use of a roulette wheel. These previous stages are performed repeatedly until the maximum number of iterations (Maxiter) reached and the optimal features are defined.

The proposed enhanced red deer optimizer (ERD)

The main idea in the proposed ERD is a tinkering strategy that enhances the traditional mating phase in the conventional red deer optimizer represented in Eq. 13. The tinkering strategy is about adding members from the worst harems, i.e.,

with the worst fitting values, toward the "best-so-far" harems which have the best fitting value. This tinkering strategy is performed based on the degree of diversity among harems.

The tinkering strategy For the tinkering strategy, we need to define the $\aleph_{harem}^{best}$ best harems and the worst $\aleph_{harem}^{worst}$ harems, according to the fitting value, while keeping $\aleph_{harem}^{best} > \aleph_{harem}^{worst}$.. The worst of the worst hinds of the U group will tinker the best harems Q, i.e., they join the original hinds. However, this tinkering strategy may lead to local minima entrapment. Accordingly, we need to control the targeted number of worst harems, $\aleph_{harem}^{worst}$, as follows.

$$\aleph_{harem}^{worst} = \aleph_{harem}^{Total} \max\left(0.5, \frac{f_{harem}^{worst}}{f_{harem}^{best}}\right) - round\left(\frac{\ddot{\imath}}{maxiter}\left(\aleph_{harem}^{Total} - \hat{\eta}\right)\right) \tag{14}$$

where $\aleph_{harem}^{Total}$ is the total number of constructed harems. F_{harem}^{worst}, and F_{harem}^{best} denote the whole worst and best fitting value, respectively. $\ddot{\imath}$ is the current iteration number corresponding to the maximum assigned number of iterations, $maxiter$. $\hat{\eta}$ is a fixed number of harems to be updated within each iteration.

After determining the worst harems, the best harems are assigned as $\aleph_{harem}^{best} = \aleph_{harem}^{worst} + 3$.. Then, the number of hinds from the worst harems are distributed equally, in number, between the best harems, while the best of the best harems assigned the worst hinds in fitting simulating the effect of mutation in metaheuristic algorithms. Now, the harems are reformulated according to the tinkering strategy. According to the mating phase, the new offspring RDs, G_{OS}, are generated as

$$G_{\mathrm{OS}} = \begin{cases} \frac{G_{com}+G_{hind}}{2} + \theta \times (u - \ell), if v > \zeta, \\ \frac{G_{com}+G_{hind}+G_{tink}}{3} + \theta \times (u - \ell), otherwise, \end{cases} \tag{15}$$

where G_{tink} is the $\xi\%$ of the new tinkering hinds. v is the diversity in fitting between the tinkered harem and the tinkering harem, as it is defined as

$$v = \left(\sum_{i=1,j=1}^{\aleph_{hind}^{harem}} \left|f_{harem_i} - f_{harem_{j\neq i}}\right|^2\right)^{1/2} \tag{16}$$

where f_{harem_i} is the whole fitting values of hinds in harem i. Algorithm 2 summarizes the main steps of the proposed ERD.

The proposed 1DCNN classification model

For the final prediction stage to differentiate between the different types of faults in the wafer map, we propose a new 1DCNN-based network which receive the resultant feature pool from the proposed ERA algorithm. 1DCNN is a specific type of CNN that is designed to basically operate on one-dimensional signal. Therefore, it employs one-dimensional convolutions and sub-sampling layers for feature mapping and extraction. Following the optimum majority of CNNs, an input layer, CNN layer group (a convolution layer and a pooling layer), a fully connected layer and output layer form a basic 1DCNN. The resultant vector from each convolutional layer, activation layer, or pooling layer can be considered as a one-dimensional "feature vector", that can be used somehow in the 1D targeted task. For the configuration of the proposed 1DCNN network, check Fig. 8.

Experimental results and discussion

In this section, different experiments have been performed to test the performance of the proposed model of wafer map fault detection. Intensive visual and computitative comparisons are introduced with an ablation study to indicate the impact of the different steps in the introduced model.

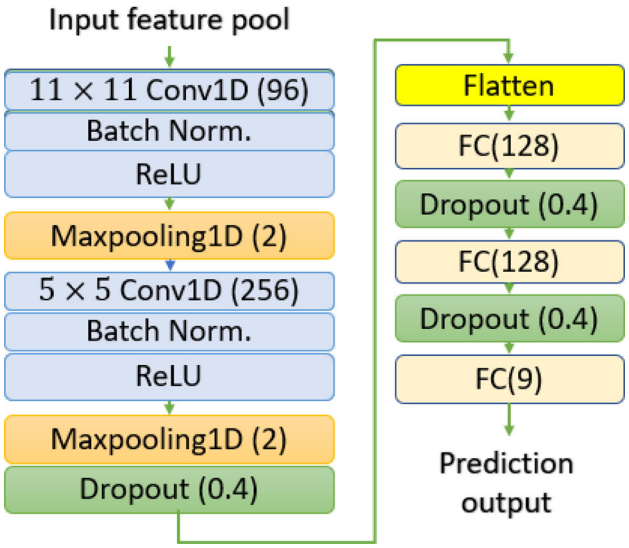


Fig. 8 The proposed 1DCNN for the final classification stage of wafer map fault types

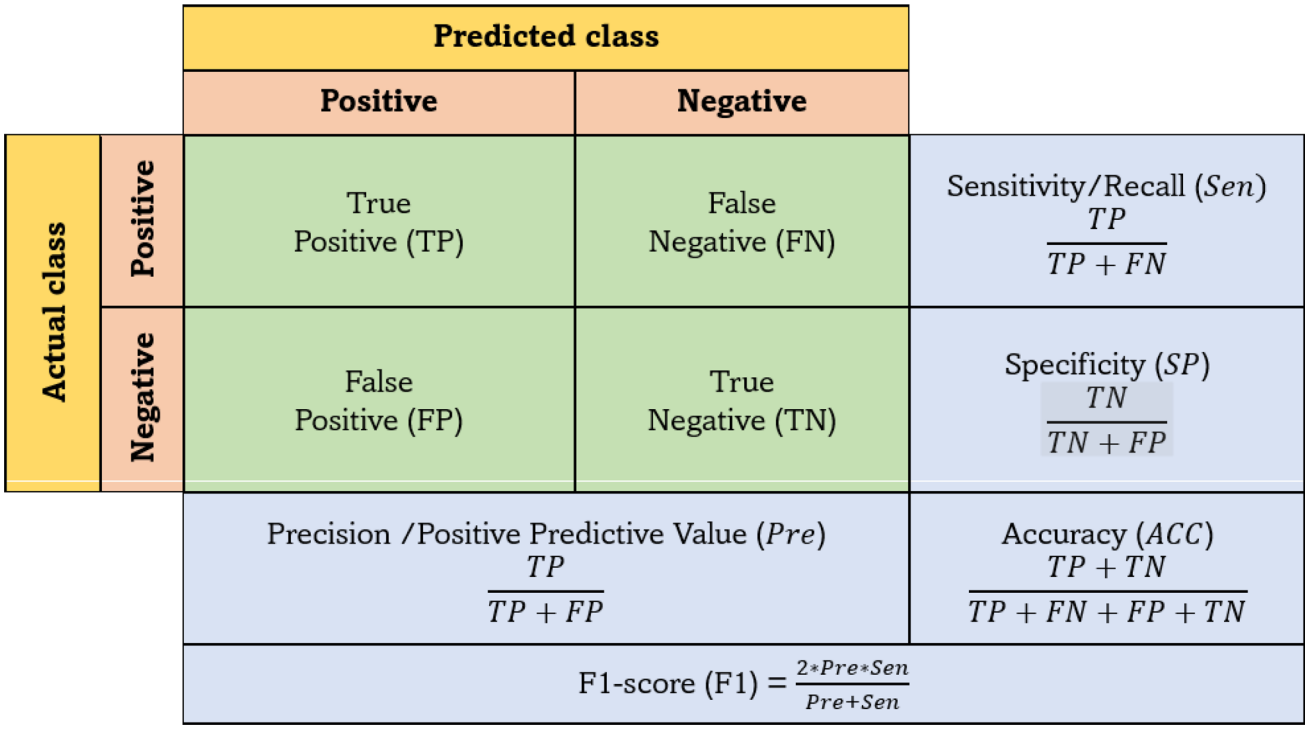


Fig. 9 The employed metrics in the evaluation process of fault detection

Experimental setup all experiments were performed on the Colab Pro environment of Google using different Python libraries, such as Keras, NumPy, Tensorflow, and Sklearn. For the feature reduction stages: check Table 3 for the configuration of SBAE and Algorithm 2 for the parameters of proposed tinkered red deer optimization (ERD). For the final prediction stage, we used the advanced optimization algorithm Adam with its default parameters to exploit the optimum weighting coefficients. We set the learning rate parameter to 0.0001.The 1DCNN is trained with batches of 32 due to memory limitations. For the model assessment, see Fig. 9, different metrices are introduced, such as accuracy, precision, recall, and F1-score.

Comparison to the state of the art

In Table 4, a comparison is set among different methods in wafer map fault detection (Jin et al., 2020; Chen et al., 2021; Yu et al., 2021a), DenseNet-GCF (Yu and Chen et al., 2022; Wang et al., 2021, 2022; Zheng et al., 2021), WDP-BNN (Alqudah et al., 2023; Zhang et al., 2022), and the proposed method. All the prementioned competitors are based on deep neural networks with supportive modules to enhance the detection performance. Most of them employ a cross-validation evaluation (Jin et al., 2020; Yu et al., 2021b, Chen et al., 2022) or just a train–test evaluation (Chen et al., 2021; Yu et al., 2021a; Xuen et al., 2022; Wang et al., 2022; Zhang et al., 2022). Following the cross-validation evaluation, Chen et al. (2022) achieved high accuracy of 98.34% for nine fault classes but with a very complicated model of two 2DCNNs employing more than 53,000 sample. On the other hand, Jin et al. (2020) declared 98.43% accuracy but with 8 fault types employing hybrid model of 2DCNN with error-correcting output codes and support vector machines for classification with 20,000 samples.

For the other types of evaluation, i.e., train–valid evaluation, it is a fragile evaluation because it can simply lead to overfitting and doesn't guarantee that the model will perform well on other unseen data from the same distribution. Wang et al. (2022) with 95:5 evaluation, for 9 fault types within 13,435 samples, declared 98.35% accuracy but with complicated model of 2DCNN and residual blocks. In addition, residual blocks can make networks more prone to overfitting. Chen et al. (2022) with 80:20 evaluation, for 8 fault types within 33,256 samples, gained 96% accuracy employing very complicated model of two 2DCNNs with error-correcting output codes and support vector machines for classification.

In Table 4, two methods, i.e., Zheng et al. (2021) and Alqudah et al. (2023), have targeted the same evaluation and feature reduction concepts as ours. Zheng et al. (2021) follows a train–validation–test evaluation (60:20:20). They declared accuracy of 93.8 with 4000 sample size for 9 fault types. Zheng et al. (2021) build their own 2DCNN for classification. Their obtained accuracy is adequate with the limited sample size and the lack of preprocessing steps. The proposed model achieves 98.77% and 99.24% accuracy for 9 and 8 fault

Table 4 Computitative comparison between the state-of-the-art methods with proposed detection method

Study	Basic techniques	Acc	Pre	Sen	F1-score	Sample size	Fault classes	Train–validation–test
Jin et al. (2020)	2DCNN–ECOC–SVM	98.43	98.2	98.3	98.2	20,000	8	10-Fold cross validation
Yu et al. (2021b)	DenseNet + GCForest	97.4	–	97.4	97.4	9112	9	5-Fold cross-validation
Chen et al. (2021)	Dual-channel 2DCNN + ECOC–SVM	96.4	96.4	96.2	96.3	33,256	8	80:20
Zheng et al. (2021)	2D-DCNN	93.8	93.8	93.8	93.8	4000	9	60:20:20
Yu et al., (2021a)	Wasserstein adversarial network	87.32	89.74	86.50	87.33	1800	9	90:10
Chen et al. (2022)	A dual-source 2D-DCNN + information entropy	98.34	93.30	89.10	90.71	53,925	9	10-Fold cross validation
Wang et al. (2022)	2D-CNN + residual blocks	98.35	98.32	98.32	98.31	13,435	9	95:5
Zhang et al. (2022)	2D BNN	98.68	92.53	94.00	93.23	31,500	9	90:10
Alqudah et al. (2023)	59 Features + SVM	82.7	83.7	84.4	84.1	29,043	8	75:25
Ours1	SBAE + ERD (15 features) + 1DCNN	**98.77**	**98.78**	**98.89**	**98.89**	18,736	9	**60:20:20**
Ours2		**98.60**	**98.67**	**98.78**	**98.8**			**36:15:49**
Ours3		**98.50**	**98.56**	**98.56**	**98.44**			**23:15:62**
Ours1	SBAE + ERD (19 features) + 1DCNN	**99.24**	**99.25**	**99.25**	**99.38**	16,397	8	**60:20:20**
Ours2		**98.66**	**0.99**	**98.75**	**98.75**			**36:15:49**
Ours3		**98.61**	**98.75**	**98.63**	**98.63**			**23:15:62**

Bold font is used to emphasize the best results

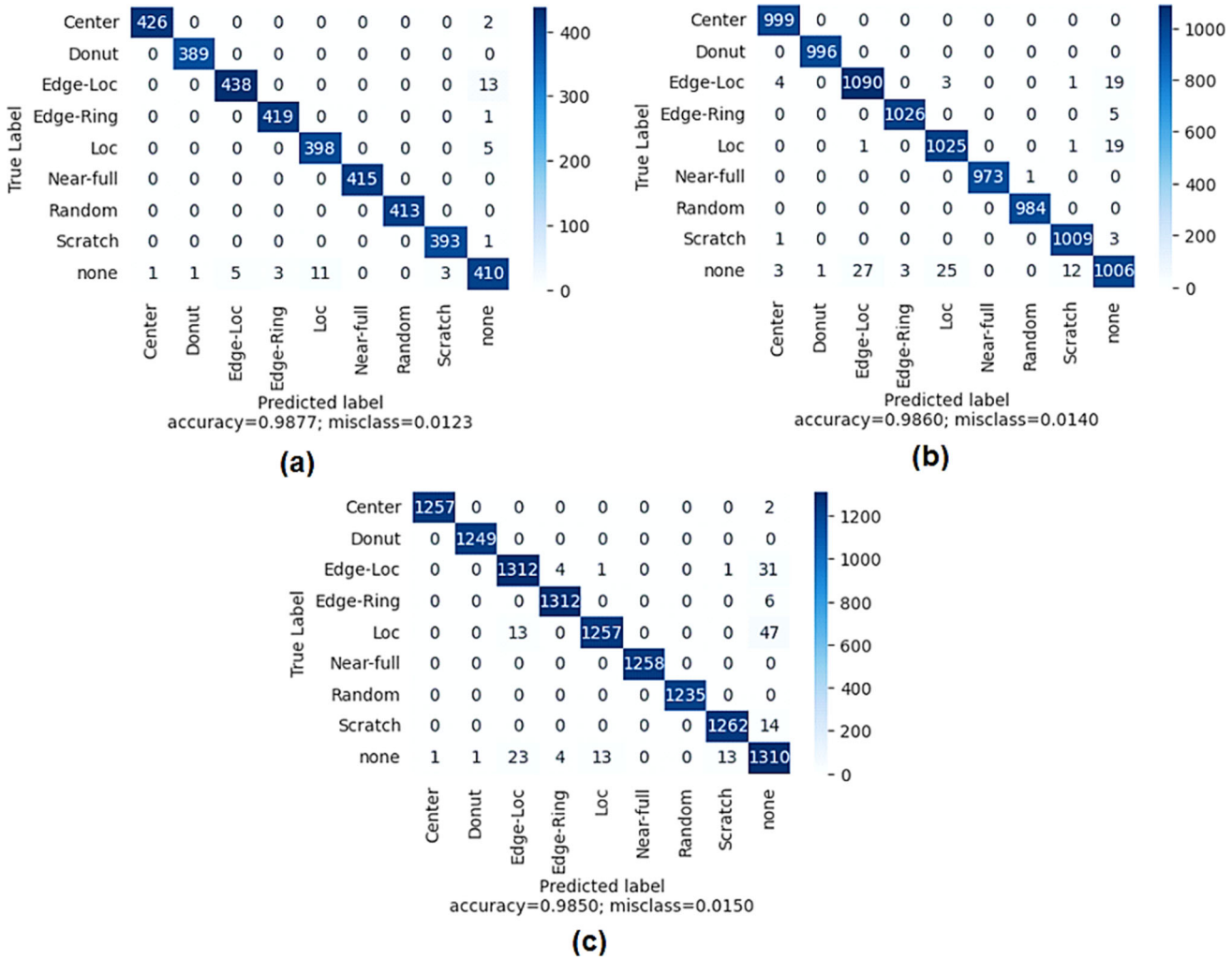


Fig. 10 The confusion matrices of the proposed wafer map fault detection model for 60:20:20 evaluation (ours1) in **a**, 36:15:49 evaluation (ours2) in **b**, and 23:15:62 evaluation (ours3) in **c**

types, respectively, with 60:20:20 evaluation within around 16,000:18,000 samples.

Alqudah et al. (2023) have followed the concept of feature reduction. Their features are extracted from the 2D wafer maps contrary to the proposed method which are from 1D sinograms. Alqudah et al. (2023) have utilized 59 feature bases in total (Density, radon, and geometry-based features). They declared accuracy of 82.7%, despite using around 29,000 sample, because they have employed a simple SVM classifier. In the proposed fault detection method, three different Train–validation–test evaluations have been introduced, i.e., 60:20:20, 36:15:49, and 23:15:62. We have got a minimum accuracy of 98.5% when the detection model is tested with 62% unseen data for 9 fault types within around 18,000 sample and 15 features. On the other side, we have gained accuracy 98.61% for 62% unseen part for 8 fault types within around 16,000 sample and 19 features. In addition, with the proposed method, the accuracy just changed from 98.77 to 98.50% and from 99.24 to 98.61% when the unseen testing data part changed from 20 to 62% for 9 and 8 fault types, respectively. This very small change proves that the proposed model generalized well by exploiting the most suitable discriminative features by the assigned feature engineering steps (SBAE and ERD). For detailed classification reports, confusion matrices, and train–validation performance of the proposed detection method for 9 fault types, check Figs.10, 11, and 12. From Figs. 10 and 11 we can see that the "none" fault type has the least metrics. It is the most confusing class. It shares similar structures (features) with other faults, check the fault images in Table 2. In addition, "none" fault is totally unbalanced with other faults, so many studies (Alqudah et al., 2023; Chen et al., 2021; Jin et al., 2020) have ignored this fault to avoid additional preprocessing steps. Figure 12 indicates the train–validation performance which shows semi-identical performance that refuting the chance of overfitting.

	precision	recall	f1-score	support
Center	1.00	1.00	1.00	428
Donut	1.00	1.00	1.00	389
Edge-Loc	0.99	0.97	0.98	451
Edge-Ring	0.99	1.00	1.00	420
Loc	0.97	0.99	0.98	403
Near-full	1.00	1.00	1.00	415
Random	1.00	1.00	1.00	413
Scratch	0.99	1.00	0.99	394
none	0.95	0.94	0.95	434
accuracy			0.99	3747
macro avg	0.99	0.99	0.99	3747
weighted avg	0.99	0.99	0.99	3747

(a)

	precision	recall	f1-score	support
Center	0.99	1.00	1.00	999
Donut	1.00	1.00	1.00	996
Edge-Loc	0.97	0.98	0.98	1117
Edge-Ring	1.00	1.00	1.00	1031
Loc	0.97	0.98	0.98	1046
Near-full	1.00	1.00	1.00	974
Random	1.00	1.00	1.00	984
Scratch	0.99	1.00	0.99	1013
none	0.96	0.93	0.95	1077
accuracy			0.99	9237
macro avg	0.99	0.99	0.99	9237
weighted avg	0.99	0.99	0.99	9237

(b)

	precision	recall	f1-score	support
Center	1.00	1.00	1.00	1259
Donut	1.00	1.00	1.00	1249
Edge-Loc	0.97	0.97	0.97	1349
Edge-Ring	0.99	1.00	0.99	1318
Loc	0.99	0.95	0.97	1317
Near-full	1.00	1.00	1.00	1258
Random	1.00	1.00	1.00	1235
Scratch	0.99	0.99	0.99	1276
none	0.93	0.96	0.94	1365
accuracy			0.99	11626
macro avg	0.99	0.99	0.99	11626
weighted avg	0.99	0.99	0.99	11626

(c)

Fig. 11 The corresponding classification reports of the confusion matrices in Fig. 10 of the proposed wafer map fault detection model for 60:20:20 evaluation (ours1) in **a**, 36:15:49 evaluation (ours2) in **b**, and 23:15:62 evaluation (ours3) in **c**

The impact of wafer map resolution in performance

WM-811K dataset originally has different sizes of wafer maps, 632 in total, and varies in resolution from 6×21 to 300×202. Figure 13 indicates the resolution distribution in terms of the number of wafer maps. There are 5 resolutions only with more than 10,000 samples. The largest resolution among them 39×37. As the wafer maps are not rich in color, i.e., each map contains only three colors, the initial preprocessing steps to adjust the size can simply affect the fault type. Hence, different experiments have been performed on the resolutions that have more than 8000 samples and they are only 7 different resolutions. These samples for each resolution are indicated in Table 5 before and after (W/O and W/) the CAE-based synthetization model as a balancing model. The "All" wafer map set groups the seven other sets and mapped them to resolution 26×26. As shown, in Table 5, the "none" fault type has the largest number of maps in each resolution. In addition, the resolution 33×29 is the most challenging as it has the least number of samples in each fault type.

Having different resolutions affect the structure of the fault types, and accordingly affect the discriminative features that affect the recognition rate of each class and thus affect the final metrics. Table 6 indicates the performance of different resolutions under different split ratios. As demonstrated, there are two resolutions that have only 8 fault types or classes, i.e., 25×27 and 27×25, but they don't have the same types of faults. The resolution 25×27 misses the fault "donut", while the resolution 27×25 misses the fault "Edge-ring", revise Table 4. The other resolutions have 9 fault types. For the category of 8 classes, the resolution 27×25 shows better performance than 25×27. With only 19 features, the resolution 27×25 provides 99.24% accuracy with 20% testing and 98.61% accuracy with 62% testing corresponding to 98.36% and 94.82% for 25×27 resolution with 22 features. We can see that the accuracy changed slightly from 99.24 to 98.61% (very good generalization) in 27×25

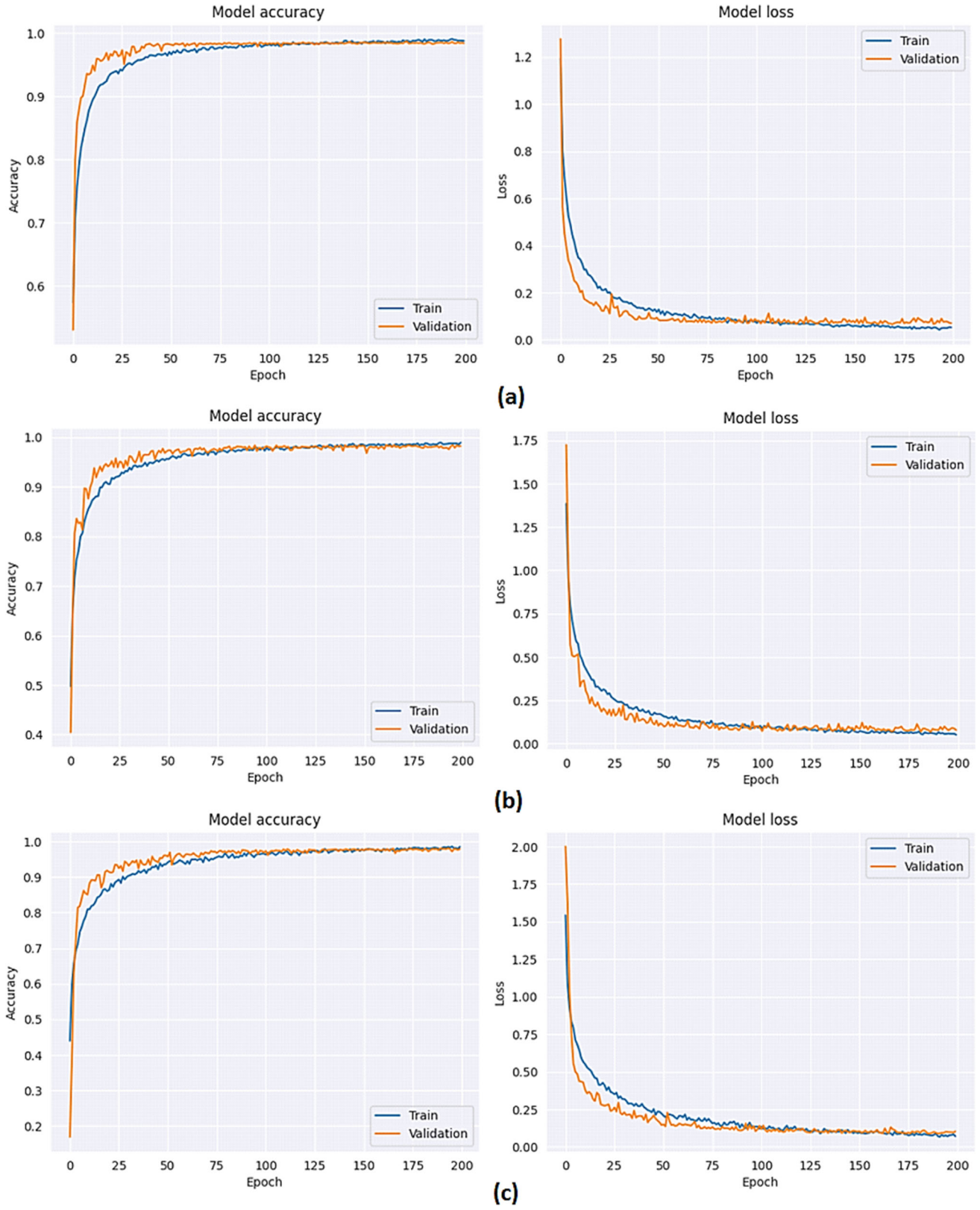


Fig. 12 Train–validation performance over epochs of the proposed wafer map fault detection in terms of accuracy and losses for 60:20:20 evaluation (ours1) in **a**, 36:15:49 evaluation (ours2) in **b**, and 23:15:62 evaluation (ours3) in **c**

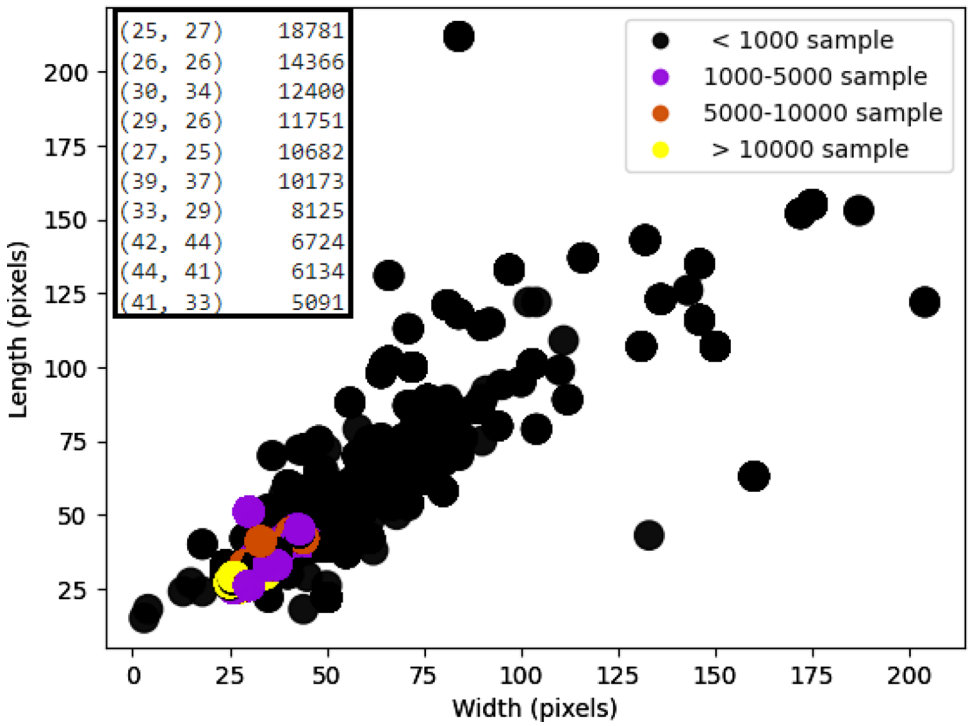


Fig. 13 the maps resolution distribution in WM-811K

and aggressively from 98.36 to 94.82% (good generalization) in 25 × 27 by changing the unseen part from 20 to 62%. As indicated from the classification reports in Fig. 14, the recognition metrics of "Loc" and "none" classes degrade a little at resolution 27 × 25. At the other side, at resolution 25 × 27, the recognition metrics of three classes degrades aggressively, i.e., "none", "loc", and "Edge-Loc", while the recognition of the "center" degrades slightly. The main cause behind a degradation in recognition metrics, which affects the grouped performance, is that the extracted discriminative feature fails to recognize this class effectively like the other classes because of sharing similar structures. Check Fig. 15 for samples of wafers for "none", "loc", "Edge-Loc", and "center".

For the resolution sets that have 9 fault types, as presented in Table 6, the resolution 33 × 29 comes at the first place with the minimum feature size (15), and with the best performance in terms of the recognition metrices, despite the variety of the split ratio. It proves a good generalization by changing the unseen part from 20 to 62%. The resolution 39 × 37 is the worst with the largest feature size (27), the least metrices, and the least generalization. Figure 16 indicates the classification reports of the resolutions 33 × 29 and 39 × 37 at different unseen part percentages, i.e., 20% and 62%. As demonstrated, the recognition metrics of "none", "Loc", and "Edge-Loc" at the resolution set 33 × 29 are high which means that the faults shapes discriminatively differ from each other, on contrary to what we have with the resolution set 39 × 37. This means the recognition rate of the fault type is independent of higher-resolution maps, but the fault type structure. In other words, the resolution set 39 × 37 has more complicated and indiscriminative fault types that are not efficiently recognizable, like the resolution set 33 × 29. For more results of other resolution sets, see Table 10 in "Appendix".

The impact of feature engineering steps

In the proposed fault detection method, two main techniques have been used for feature engineering after the data balancing step. The first is a sparse feature learning and encoding by the proposed SBAE, where the resultant sparse encoded feature maps are extracted from the bottleneck of SBAE. Inducing sparse regularization in a traditional convolutional autoencoder enhances the reconstruction process and accordingly efficient sparse embedding can be gained at the bottleneck of SBAE. In Fig. 17 a comparison is made between the performance of SBAE and the traditional convolutional AE (CAE) that have the same configuration excluding the sparsity regularization. As shown, without the induced sparsity, the reconstructed wafer maps at CAE are blurry without any fine details contrary to SBAE. The second feature engineering step is applying the proposed tinkered red deer feature ranking (ERD) to 1D sinograms of the resultant sparse encoded features from the bottleneck of SBAE. Figure 18 indicates visual comparison for the convergence of fitness over iterations between the conventional red deer

Table 5 The number of samples under each employed wafer map resolution, from the labeled maps, before and after the CAE-based synthetization model

Fault type	26 × 26		25 × 27		29 × 26		27 × 25		30 × 34		39 × 37		33 × 29		All	
	W/O CAE	W/ CAE	W/O CAE	W/ CAE	W/O CAE	W/ CAE	W/O CAE	W/ CAE	W/O CAE	W/ CAE	W/O CAE	W/ CAE	W/O CAE	W/ CAE	W/O CAE	W/ CAE
Center	90	2160	2251	2251	50	2100	23	2024	58	2088	173	2249	20	2040	2665	3475
Donut	1	2002	–	–	2	2004	6	2010	4	2008	22	2024	2	2004	37	3524
Edge-Loc	296	2368	355	2485	233	2330	96	2112	306	2448	473	2838	130	2210	1889	3563
Edge-Ring	31	2046	25	2050	64	2112	–	–	9	2016	15	2025	55	2090	199	3480
Loc	297	2376	172	2236	117	2223	104	2184	241	2410	222	2442	107	2140	1260	3679
Near-full	16	2032	21	2037	5	2010	11	2013	5	2010	15	2025	28	2044	101	3612
Random	74	2146	53	2067	4	2008	10	2020	18	2034	35	2065	12	2016	206	3662
Scratch	72	2088	23	2024	80	2160	7	2009	69	2070	60	2100	27	2068	358	3873
None	13,489	2489	15,881	2181	11,196	2196	10,425	2025	11,690	2090	9158	2158	7724	2124	79,563	3604
$\sum$	14,366	**19,707**	18,781	**17,331**	11,751	**19,143**	10,682	**16,397**	12,400	**19,174**	10,173	**19,926**	8105	**18,736**	86,278	**32,472**

Bold font is used to emphasize the best results

Table 6 Computitative comparison between the performance of different resolutions of wafer maps at different split ratio

Wafer map resolution	Final feature size/classes	Acc	Pre	Sen	F1-score	Sample size	Train–valid–test
25 × 27	22/8	98.36	98.37	98.37	98.5	17,331	60:20:20
		96.83	96.88	96.88	97		36:15:49
		94.82	95.25	94.88	94.88		23:15:62
27 × 25	19/8	99.24	99.25	**99.25**	99.38	16,397	60:20:20
		98.66	0.99	98.75	98.75		36:15:49
		98.61	98.75	98.63	98.63		23:15:62
39 × 37	27/9	97.37	97.89	97.22	97.33	19,926	60:20:20
		96.17	96.56	96.22	96.33		36:15:49
		93.07	93.33	93.56	93.22		23:15:62
33 × 29	15/9	98.77	98.78	**98.89**	98.89	18,736	60:20:20
		98.60	**98.67**	**98.78**	**98.8**		36:15:49
		98.50	**98.56**	**98.56**	**98.44**		23:15:62

Bold font is used to emphasize the best results

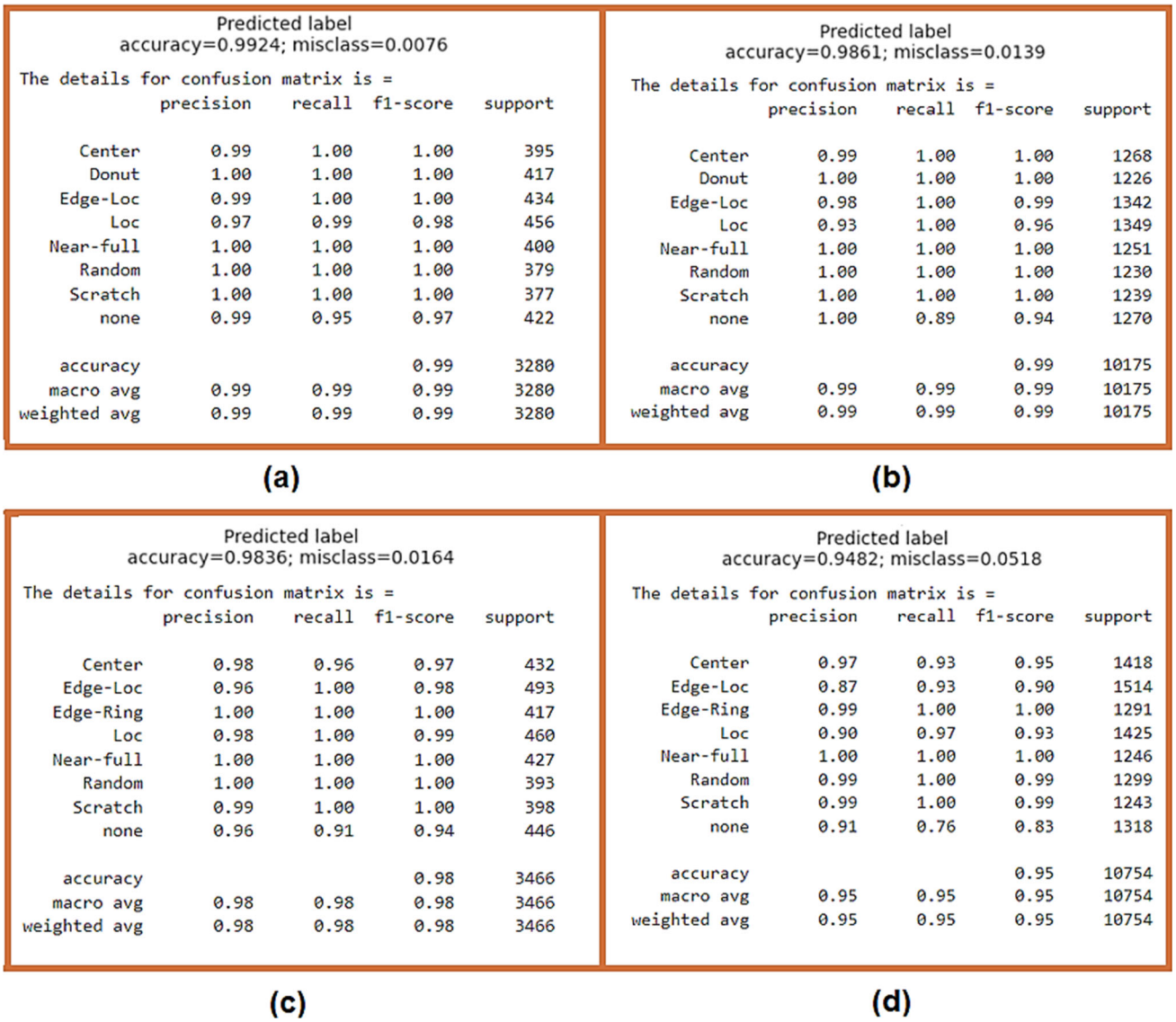


Fig. 14 Visual comparison between different resolutions of wafer maps in terms of the classification report, 27 × 25 (8 classes) in the first row and 25 × 27 (8 classes) in the second row. The first column at 60:20:20 while the second at 23:15:62 (train–validation–test) evaluation

optimization and the proposed ERD. Better convergence and higher fitness belongs to the proposed ERD.

The impact of these prementioned feature engineering steps on performance can be checked through four conditions: the first is without any feature engineering steps, the second and the third adopt only one feature engineering step, i.e., SBAE or ERD, the fourth is about applying both feature engineering steps. These conditions are presented in Table 7 as computitative comparison. As shown, the worst performance belongs to the first condition (W/O ERD, W/O SBAE), despite owning the largest 1D feature pool. The redundant information in this large pool limits the performance of the proposed classifier 1DCNN. We can see that the most wafer maps affected by the absence of the feature engineering steps are the set of "33 × 29", while the least affected are the set of "30 × 34". This means that the structure of fault types in the set 33 × 29 offer more redundant information than that of the set "30 × 34". For the second and the third condition of applying one feature engineering step, i.e., (W/ ERD, W/O SBAE) and (W/O ERD, W/ SBAE), respectively, better performance than the first condition is gained with smaller 1D feature pool.

As pointed out, in (W/ ERD, W/O SBAE), considering the original features without the sparse learning by SBAE increases the number of features selected by the proposed ERD algorithm seeking the global optimum feature ranking. The set "30 × 34" has the largest feature pool but with high accuracy, while the set "33 × 29" has the smallest feature pool but with low accuracy. On the other hand, depending on the sparse boosted features from the bottleneck of SBAE without

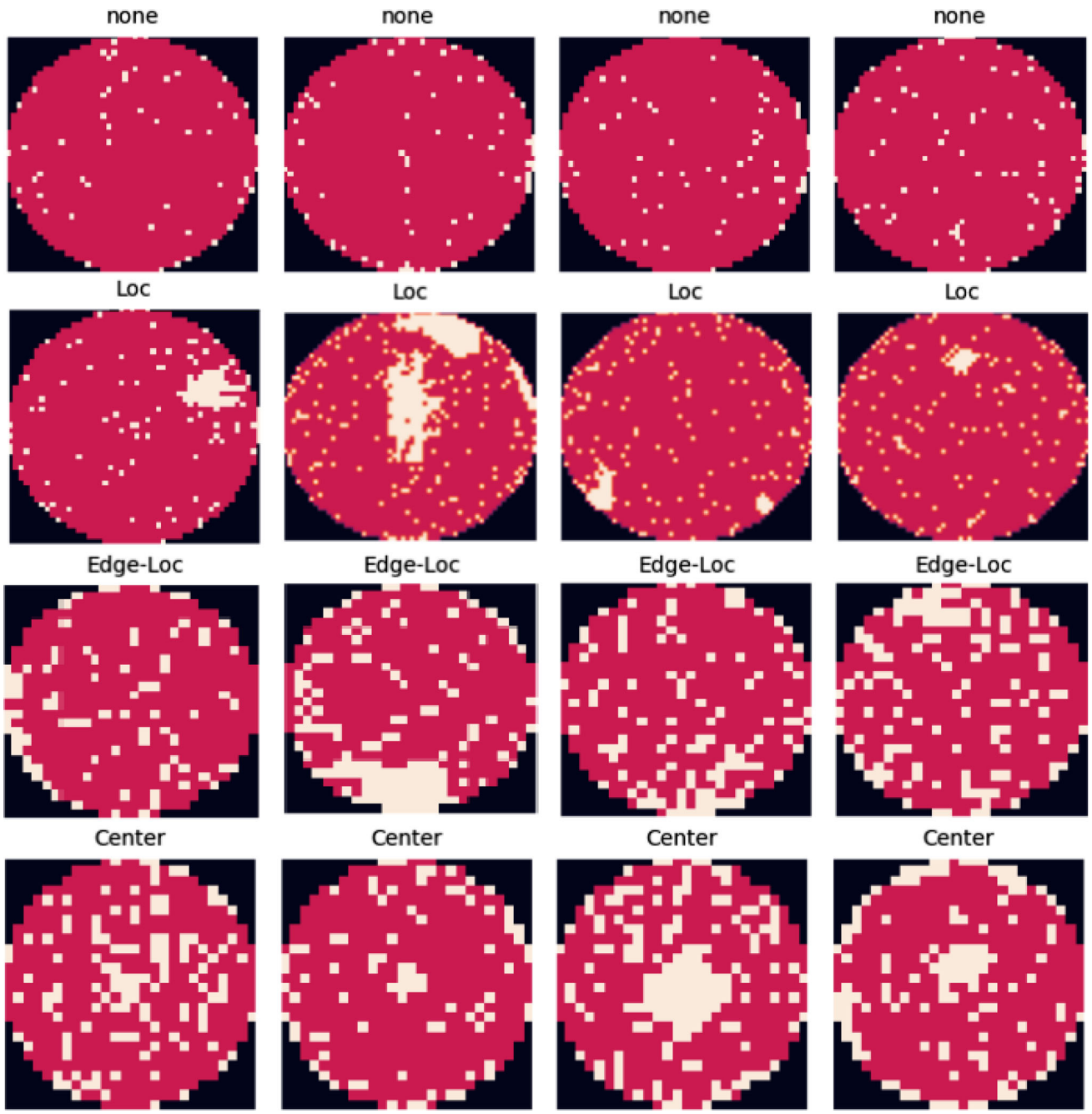
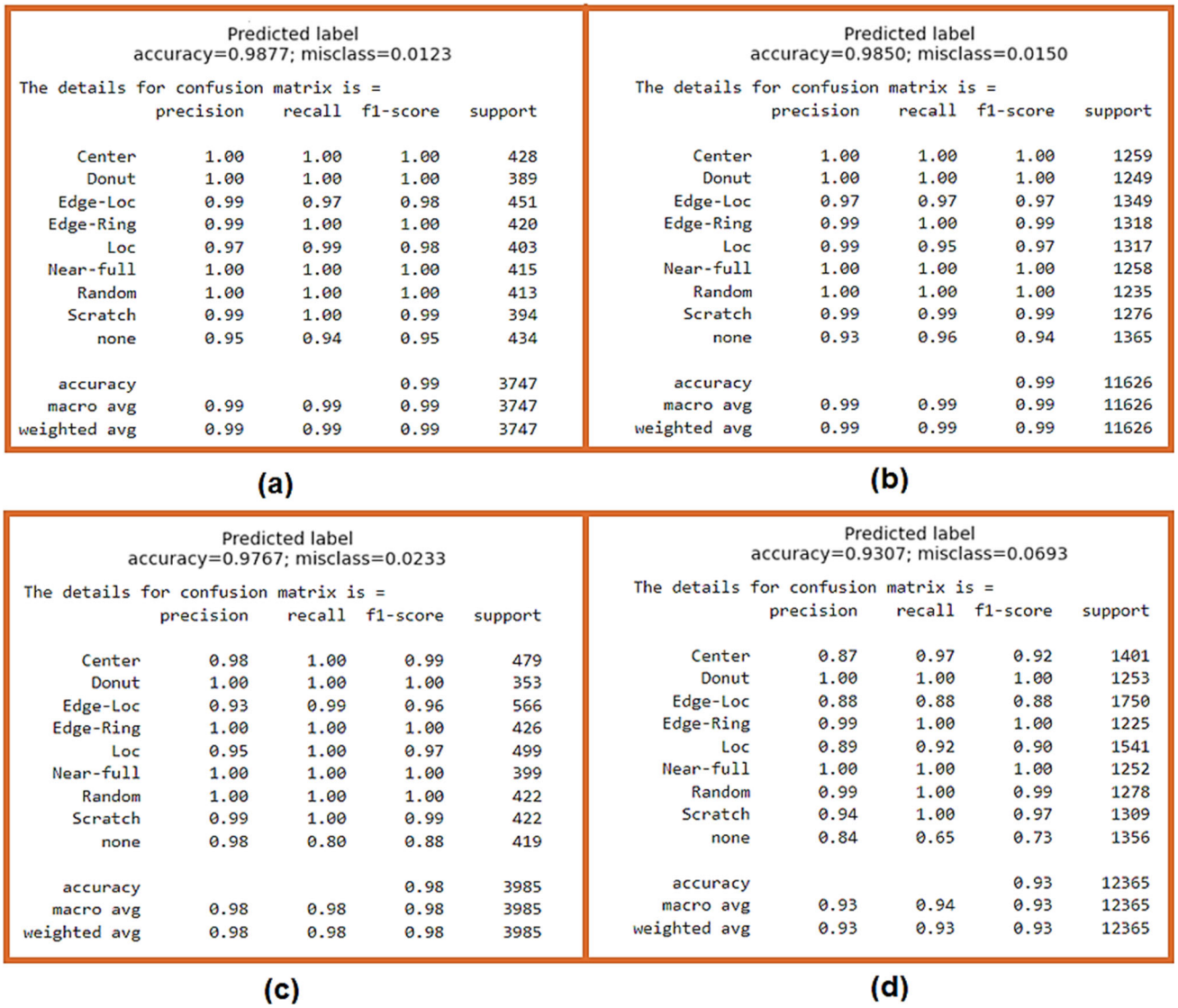


Fig. 15 Samples of wafer maps from fault types: "none", "Loc", "Edge-Loc", and "Center"

employing ERD at the third condition (W/O ERD, W/ SBAE) provides a fixed size 1D feature pool which shows adequate performance with some sets, like 27×25 and 33×29, but fails with other sets, like 26×26, 30×34. The adequate performance means that the extracted features guarantees a suitable discrimination between classes while failing means that features are not discriminative enough. Finally, at the fourth condition (W/ ERD, W/ SBAE), as signified, the least 1D feature pool sizes with the best performance is achieved. The sparsity induced in SBAE allows to get more sparse fine details that simplifies the mission of ERD to find the global optimum with the least number of features. For more computitative results of the four conditions at other different resolutions, see Table 11 in "Appendix". Figure 19 presents the classification reports of the map set "26×26" at the previous four conditions. In addition, Fig. 20 indicates the train–validation performance over epochs. The small gap between train and valid means good generalization while large gap means bad generalization.



Predicted label
accuracy=0.9877; misclass=0.0123

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	1.00	1.00	1.00	428
Donut	1.00	1.00	1.00	389
Edge-Loc	0.99	0.97	0.98	451
Edge-Ring	0.99	1.00	1.00	420
Loc	0.97	0.99	0.98	403
Near-full	1.00	1.00	1.00	415
Random	1.00	1.00	1.00	413
Scratch	0.99	1.00	0.99	394
none	0.95	0.94	0.95	434
accuracy			0.99	3747
macro avg	0.99	0.99	0.99	3747
weighted avg	0.99	0.99	0.99	3747

(a)

Predicted label
accuracy=0.9850; misclass=0.0150

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	1.00	1.00	1.00	1259
Donut	1.00	1.00	1.00	1249
Edge-Loc	0.97	0.97	0.97	1349
Edge-Ring	0.99	1.00	0.99	1318
Loc	0.99	0.95	0.97	1317
Near-full	1.00	1.00	1.00	1258
Random	1.00	1.00	1.00	1235
Scratch	0.99	0.99	0.99	1276
none	0.93	0.96	0.94	1365
accuracy			0.99	11626
macro avg	0.99	0.99	0.99	11626
weighted avg	0.99	0.99	0.99	11626

(b)

Predicted label
accuracy=0.9767; misclass=0.0233

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	0.98	1.00	0.99	479
Donut	1.00	1.00	1.00	353
Edge-Loc	0.93	0.99	0.96	566
Edge-Ring	1.00	1.00	1.00	426
Loc	0.95	1.00	0.97	499
Near-full	1.00	1.00	1.00	399
Random	1.00	1.00	1.00	422
Scratch	0.99	1.00	0.99	422
none	0.98	0.80	0.88	419
accuracy			0.98	3985
macro avg	0.98	0.98	0.98	3985
weighted avg	0.98	0.98	0.98	3985

(c)

Predicted label
accuracy=0.9307; misclass=0.0693

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	0.87	0.97	0.92	1401
Donut	1.00	1.00	1.00	1253
Edge-Loc	0.88	0.88	0.88	1750
Edge-Ring	0.99	1.00	1.00	1225
Loc	0.89	0.92	0.90	1541
Near-full	1.00	1.00	1.00	1252
Random	0.99	1.00	0.99	1278
Scratch	0.94	1.00	0.97	1309
none	0.84	0.65	0.73	1356
accuracy			0.93	12365
macro avg	0.93	0.94	0.93	12365
weighted avg	0.93	0.93	0.93	12365

(d)

Fig. 16 Visual comparison between different resolutions of wafer maps in terms of the classification report, 33×29 (9 classes, 15 features) in the first row and 39×37 (9 classes, 27 features) in the second row. The first column at 60:20:20 while the second at 23:15:62 (train–validation–test) evaluation

The impact of the proposed ERD compared to other metaheuristic algorithms.

Here, the impact of the enhanced tinkered red deer algorithm (ERD) is discussed. A comparison is set in Table 8 between different metaheuristics algorithms compared to the proposed ERD. A metaheuristic algorithm (Abdel-Basset et al., 2018) is a high-level procedure or heuristic designed to find, generate, tune, or select a heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem. In the prementioned comparison, Genetic (GA; Sohail, 2023), Equilibrium (EO; Altantawy & Kishk, 2023; Houssein et al., 2022), Grey Wolf (GWO; Faris et al., 2018), Sine cosine (SCA; Zhou et al., 2022), and particle swarm algorithms (PSO; Shami et al., 2022) have been utilized. As demonstrated from Table 8, all assigned metaheuristic algorithms provide a superior performance in the fault type prediction, but the proposed ERD provides the absolute least feature pool size with the same accuracy, approximately. For the 8-fault type detection problem, the average drop in accuracy from the best performer is 0.63%, while in 9-fault type problem, the average drop in accuracy is 1.22%. In the set of "All", it is noticed that the accuracy degraded a little bit compared to other sets, because of the resizing process of wafer map to a fixed common size which affects the shape of fault patterns. Table 8 indicates the results

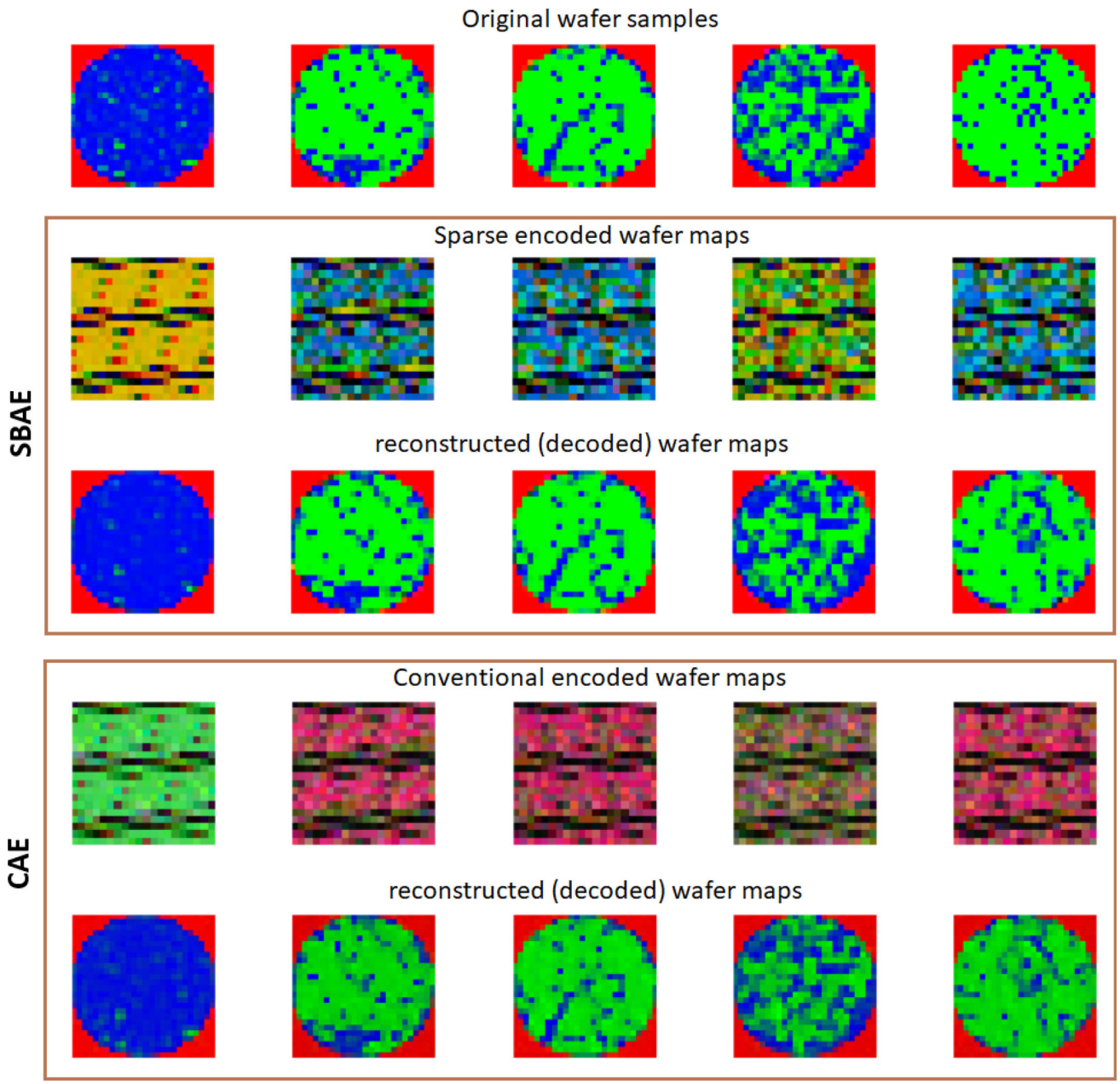


Fig. 17 Comparison between the performance of SBAE and CAE

for the resolution sets: 26 × 26, 27 × 25, and All. For more results of other resolution sets, see Table 12 in "Appendix". For fair comparison in Tables 8 and 12, we have utilized the same population size and seek the parameters that keep the most possible fitting score on all employed metaheuristic algorithms.

The main cause that the proposed ERD selects a smaller number of features is generally the tendency to focus more on exploitation than other metaheuristic algorithms, such as GA, PSO, EO, GWO, and SCA. The tendency of emphasizing on exploitation rises from the mating behavior of red deer, where the dominant stag (leader) mates with the most fertile hinds. This mechanism symbolizes the selection of features with higher fitness, driving the algorithm towards refining a smaller set of relevant features. In contrast, algorithms like GA and PSO tend to explore the search space more extensively, potentially leading to the selection of a larger number of features. This is because their mechanisms encourage the exploration of diverse solutions and the recombination of features, which can sometimes introduce less relevant features into the selected set. While ERD focus on exploitation can be beneficial in reducing the risk of overfitting, it may

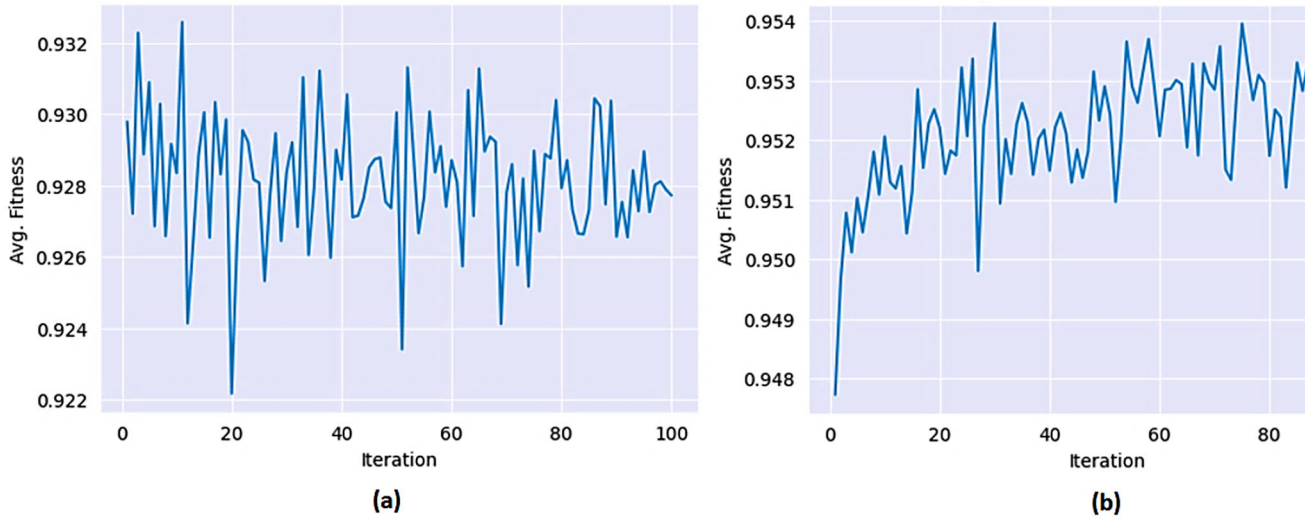


Fig. 18 The convergence of fitness over iterations for the traditional red deer in **a** and the tinkered version in **b**

Table 7 Comparison between the different cases of applying the adopted feature engineering steps, i.e., ERD and SBAE

Tested condition	Resolution	Final feature size/classes	Acc	Pre	Sen	F1-score
W/O ERD W/O SBAE	26 × 26	676/9	76.59	84.67	75.78	75.11
	27 × 25	675/8	85.49	80.25	86.87	82.62
	30 × 34	1020/9	95.57	96.22	95.67	96
	33 × 29	957/9	73.23	0.78	73.33	0.71
W/ ERD W/O SBAE	26 × 26	73/9	98.05	98.22	98.22	98.22
	27 × 25	37/8	99.12	99.25	99.13	99.25
	30 × 34	280/9	97.57	97.67	97.78	97.67
	33 × 29	28/9	94.98	0.95	94.89	94.89
W/O ERD W/ SBAE	26 × 26	169/9	93.61	95.33	94.33	94.11
	27 × 25	169/8	**99.48**	**99.63**	**99.38**	**99.5**
	30 × 34	169/9	90.33	91.78	90.22	90.22
	33 × 29	169/9	**99.04**	**99.11**	**99.11**	**99.11**
W/ ERD W/ SBAE	26 × 26	22/9	**98.71**	**98.78**	**98.89**	**98.89**
	27 × 25	19/8	99.24	99.25	99.25	99.38
	30 × 34	23/9	**98.17**	**98.44**	**98.11**	**98.11**
	33 × 29	15/9	98.78	98.89	98.89	98.77

Bold font is used to emphasize the best results

also limit the algorithm's ability to capture complex relationships between features. This could potentially affect its performance on datasets where such relationships are crucial for accurate prediction or classification.

The impact of the classification stage

For the prediction stage, different common 1D deep networks (Kiranyaz et al., 2021) have been tested, like 1D-VGG16, 1D-ResNet50, 1D-LeNet-5, and 1D-Inception, against the proposed 1DCNN. A comparison is set at Table 9 between these networks. As demonstrated, the 1D-VGG16 provides the best average accuracy of 98.27%, but with larger parameters which are three times the parameters of the proposed 1DCNN which comes in the second place with average accuracy of 98.08%. The largest parameters and the worst performance belong to 1D-ResNet50 (16 M, 96.68%). The smallest number of parameters, 16 K, belongs to 1D-Inception with

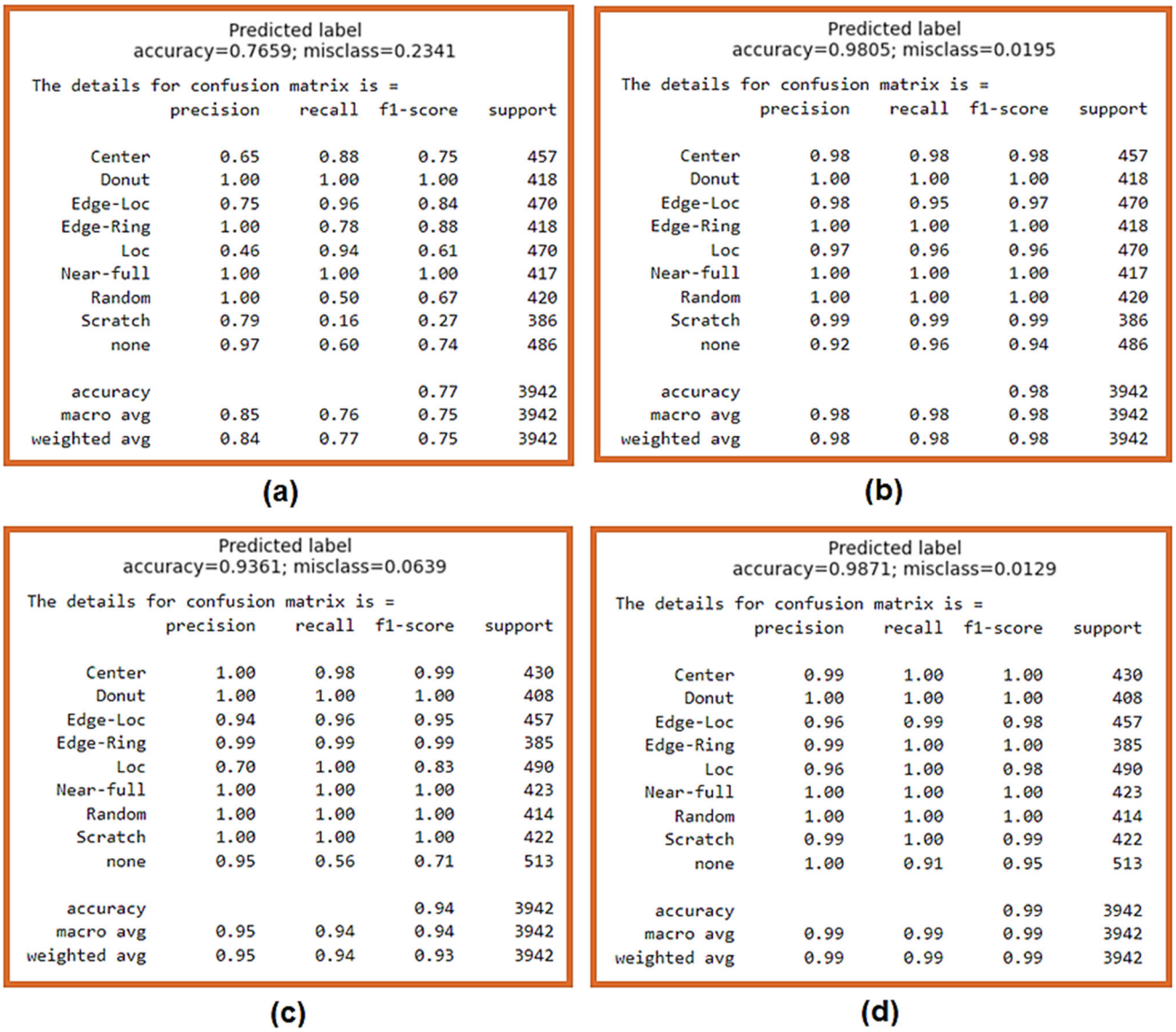


Fig. 19 Visual classification reports comparison of different conditions of applying the feature engineering steps, i.e., SBAE and ERD, to wafer maps of size 26×26; **a** W/O ERD and W/O SBAE (676 features), **b** W/ ERD and W/O SBAE (73 features), **c** W/O ERD and W/ SBAE (169 features), and **d** W/ ERD and W/ SBAE (22 features)

an average accuracy of 97.23%. The introduced 1D-LeNet-5 shows an average accuracy of 97.96% with 240 K number of parameters. For more computitative results, see Table 13 in "Appendix". Figure 21 introduces a visual comparison between the prementioned classifier in terms of the classification report. The main difference between classifiers that affect the total performance is the recognition metrics of "none", "loc", and "edge-loc", as the main difficult fault types in recognition. In Fig. 22, the training and validation losses are indicated over epochs. As shown, the proposed 1DCNN and 1D-LeNet-5 demonstrate the least losses and the smallest gap between validation and training which provides good generalization.

Conclusion

In this paper, a hybrid deep model for fault type prediction in wafer maps is proposed. The proposed model has targeted three objectives. The first is getting over the highly imbalanced dataset. The second targets obtaining more discriminative reduced features in 1D form instead of the 2D form of the original wafer maps. The final is an effective classifier that achieves adequate balance between accuracy and complexity. For the first objective, a new unsupervised synthetization model as a CAE is proposed which succeeded in reconstructing the inserted wafer maps with a very low loss of 0.0011. For achieving more discriminative features, as the

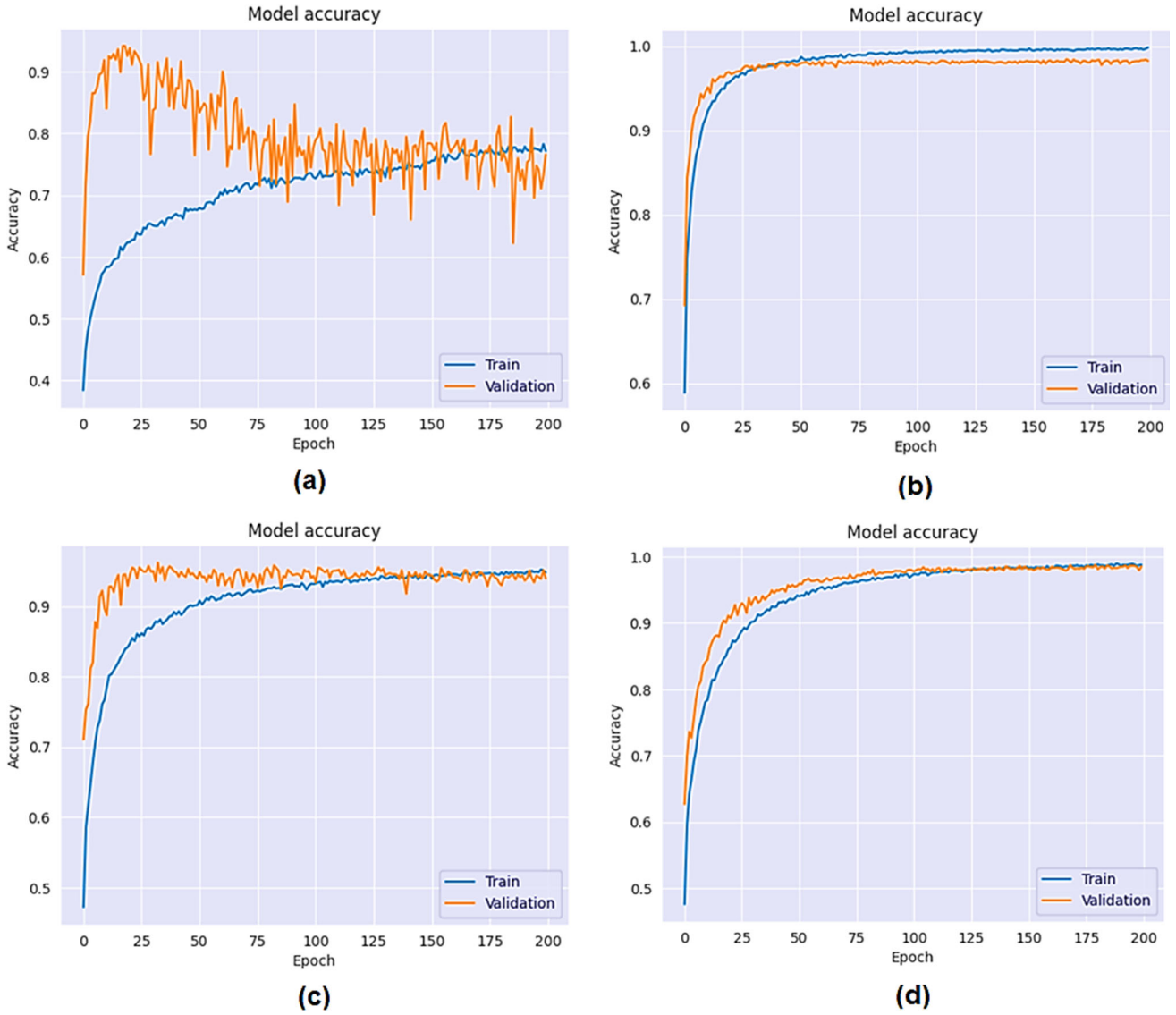


Fig. 20 Train–validation performance of different conditions of applying the feature engineering steps, i.e., SBAE and ERD, to wafer maps of size 26 × 26; **a** W/O ERD and W/O SBAE (676 features), **b** W/ ERD and W/O SBAE (73 features), **c** W/O ERD and W/ SBAE (169 features), and **d** W/ ERD and W/ SBAE (22 features)

second objective, firstly, a new sparsity-boosted autoencoder (SBAE) is proposed to get sparse encoded maps with 50% reduction in spatial size compared to the original maps. Secondly, an enhanced tinkered red deer optimization (ERD) is applied to 1D sinograms of the previously obtained sparse maps to get an average final 1D feature pool of ~ 25 feature bases (~ 1.5% of the original maps). In an ablation study, the adopted feature engineering steps prove efficiency in getting the least possible feature bases with high accuracy, especially when it is compared to other metaheuristic algorithms, such as Genetic (GA), Equilibrium (EO), Grey Wolf (GWO), Sine cosine (SCA), and particle swarm (PSO) algorithms. For the third objective, a new 1DCNN model is proposed for the 9- and 8-fault type prediction, which achieves an average accuracy of 98.1% with 180 K of trainable parameters. The proposed 1DCNN is compared to other common 1DCNNs, such as 1D-VGG16 (98.26%, 590 K), 1D-ResNet50 (96.7%, 16 M), 1D-LeNet-5 (98%, 240 K), and 1D-Inception (97.23%, 16 K). Despite the achievements of the proposed wafer map inspection model, being a hybrid deep model still increases the computational cost of the inspection procedure. Accordingly, as a future work, we intend to work on the feature engineering steps to reduce its complexity. We will try to extend the proposed detection model to other classification problems in wafer maps and other datasets, as well.

Table 8 Comparison of different common metaheuristic algorithms for the proposed fault detection in wafer maps

Wafer map resolution	Final feature size/classes	Metaheuristic algorithm	Acc	Pre	Sen	F1-score
26 × 26	179/9	GA	99.24	99.33	99.33	99.44
	229/9	EO	99.32	99.44	99.44	99.56
	155/9	SCA	99.21	99.22	99.33	99.33
	152/9	GWO	99.16	99.22	99.22	99.22
	174/9	PSO	99.11	99.11	99.22	99.11
	22/9	ERD	98.71	98.78	98.89	98.89
27 × 25	152/8	GA	**99.66**	**99.75**	99.63	99.63
	201/8	EO	99.70	99.63	**99.75**	**99.88**
	190/8	SCA	99.70	**99.75**	**99.75**	99.75
	154/8	GWO	99.70	**99.75**	**99.75**	99.75
	157/8	PSO	99.73	**99.75**	**99.75**	99.75
	19/8	ERD	99.24	99.25	99.25	99.38
All (26 × 26)	199/9	GA	97.20	97.22	97.22	97.22
	245/9	EO	97.29	97.44	97.33	97.22
	299/9	SCA	97.43	97.78	97.33	97.44
	200/9	GWO	97.21	97.33	97.22	97.22
	225/9	PSO	97.81	97.67	97.89	97.89
	45/9	ERD	95.15	95.11	95.11	95

Bold font is used to emphasize the best results

Table 9 Comparison of different common 1D CNNs for the proposed fault detection in wafer maps

Wafer map resolution	Final feature size/classes	Classification technique	Acc	Pre	Sen	F1-score	No of parameters
26 × 26	22/9	1D-VGG16	98.86	98.89	99	**99**	594 K
		1D-ResNet50	97.84	97.89	98.11	98	16 M
		1D-LeNet-5	98.48	98.67	98.67	98.67	244 K
		1D-Inception	98.20	98.33	98.56	98.33	15 K
		Our 1DCNN	98.71	**98.78**	98.89	98.89	175 K
27 × 25	19/8	1D-VGG16	**98.96**	**99.22**	99	99	561 K
		1D-ResNet50	97.93	98	98	98	16 M
		1D-LeNet-5	98.87	99	98.88	98.88	211 K
		1D-Inception	99.18	99.25	**99.25**	99.25	15 K
		Our 1DCNN	99.24	99.25	**99.25**	**99.38**	175 K
30 × 34	23/9	1D-VGG16	98.36	98.56	98.33	98.33	594 K
		1D-ResNet50	97.31	97.78	97.33	97.11	16 M
		1D-LeNet-5	97.97	98.22	97.89	97.78	244 K
		1D-Inception	97.91	98.11	97.89	97.89	16 K
		Our 1DCNN	98.17	98.44	98.11	98.11	176 K

Bold font is used to emphasize the best results

Predicted label
accuracy=0.9886; misclass=0.0114

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	0.99	1.00	0.99	430
Donut	1.00	1.00	1.00	408
Edge-Loc	0.97	0.99	0.98	457
Edge-Ring	0.99	1.00	1.00	385
Loc	0.97	1.00	0.99	490
Near-full	1.00	1.00	1.00	423
Random	1.00	1.00	1.00	414
Scratch	0.98	1.00	0.99	422
none	1.00	0.92	0.96	513
accuracy			0.99	3942
macro avg	0.99	0.99	0.99	3942
weighted avg	0.99	0.99	0.99	3942

(a)

Predicted label
accuracy=0.9784; misclass=0.0216

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	0.98	1.00	0.99	430
Donut	1.00	1.00	1.00	408
Edge-Loc	0.93	1.00	0.97	457
Edge-Ring	0.99	0.96	0.98	385
Loc	0.94	1.00	0.97	490
Near-full	1.00	1.00	1.00	423
Random	1.00	1.00	1.00	414
Scratch	0.97	1.00	0.98	422
none	1.00	0.87	0.93	513
accuracy			0.98	3942
macro avg	0.98	0.98	0.98	3942
weighted avg	0.98	0.98	0.98	3942

(b)

Predicted label
accuracy=0.9848; misclass=0.0152

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	0.97	1.00	0.99	430
Donut	1.00	1.00	1.00	408
Edge-Loc	0.98	0.99	0.98	457
Edge-Ring	0.99	1.00	1.00	385
Loc	0.96	1.00	0.98	490
Near-full	1.00	1.00	1.00	423
Random	1.00	1.00	1.00	414
Scratch	0.98	1.00	0.99	422
none	1.00	0.89	0.94	513
accuracy			0.98	3942
macro avg	0.99	0.99	0.99	3942
weighted avg	0.99	0.98	0.98	3942

(c)

Predicted label
accuracy=0.9820; misclass=0.0180

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	1.00	1.00	1.00	430
Donut	1.00	1.00	1.00	408
Edge-Loc	0.95	0.99	0.97	457
Edge-Ring	0.99	1.00	0.99	385
Loc	0.94	1.00	0.97	490
Near-full	1.00	1.00	1.00	423
Random	1.00	1.00	1.00	414
Scratch	0.98	1.00	0.99	422
none	0.99	0.88	0.93	513
accuracy			0.98	3942
macro avg	0.98	0.98	0.98	3942
weighted avg	0.98	0.98	0.98	3942

(d)

Predicted label
accuracy=0.9871; misclass=0.0129

The details for confusion matrix is =

	precision	recall	f1-score	support
Center	0.99	1.00	1.00	430
Donut	1.00	1.00	1.00	408
Edge-Loc	0.96	0.99	0.98	457
Edge-Ring	0.99	1.00	1.00	385
Loc	0.96	1.00	0.98	490
Near-full	1.00	1.00	1.00	423
Random	1.00	1.00	1.00	414
Scratch	0.99	1.00	0.99	422
none	1.00	0.91	0.95	513
accuracy			0.99	3942
macro avg	0.99	0.99	0.99	3942
weighted avg	0.99	0.99	0.99	3942

(e)

Fig. 21 Visual classification reports of different 1D deep classifier for the prediction of fault type in wafer maps with resolution of 26 × 26. **a** 1D-VGG-16, **b** 1D-ResNet50, **c** 1D-LeNet-5, **d** 1D-Inception, and **e** the proposed 1DCNN

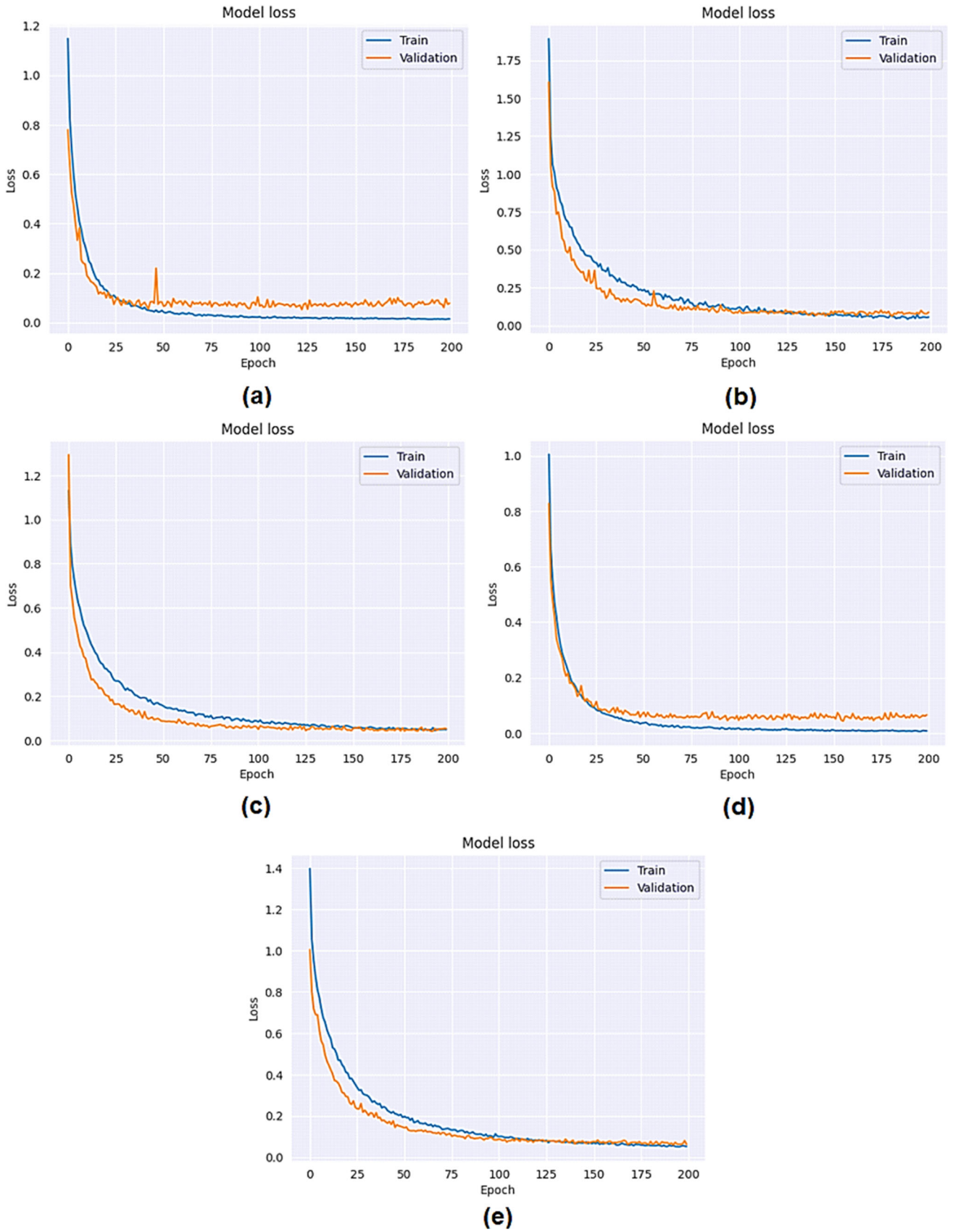


Fig. 22 Visual classification reports of different 1D deep classifier for the prediction of fault type in wafer maps with resolution of 26 × 26. **a** 1D-VGG-16, **b** 1D-ResNet50, **c** 1D-LeNet-5, **d** 1D-Inception, and **e** the proposed 1DCNN

Table 10 Computitative comparison between the performance of different resolutions of wafer maps at different split ratio

Wafer map resolution	Final feature size/classes	Acc	Pre	Sen	F1-score	Sample size	Train–valid–test
26 × 26	22/9	98.71	**98.78**	98.89	98.89	19,707	60:20:20
		96.68	96.88	97	96.78		36:15:49
		94.32	94.56	94.78	94.56		23:15:62
29 × 26	22/9	**98.85**	98.89	98.89	**99**	19,143	60:20:20
		98.37	98.44	98.56	98.33		36:15:49
		97.19	97.22	97.33	97.22		23:15:62
30 × 34	23/9	98.17	98.44	98.11	98.11	19,174	60:20:20
		97.93	98.11	98	98.22		36:15:49
		95.39	95.78	95.56	95.56		23:15:62
All (26 × 26)	45/9	95.15	95.11	95.11	95	32,472	60:20:20
		92.80	92.88	92.88	92.78		36:15:49
		89.70	89.78	89.67	89.56		23:15:62

Bold font is used to emphasize the best results

Table 11 Comparison between the different cases of applying the adopted feature engineering steps, i.e., ERD and SBAE

Tested condition	Resolution	Final feature size/classes	Acc	Pre	Sen	F1-score
W/O ERD W/O SBAE	29 × 26	754/9	87.26	81.55	87	83.22
	27 × 25	675/8	85.49	80.25	86.87	82.62
	39 × 37	1443/9	85.45	79.78	85.56	81.89
	All	676/9	80.17	82.89	79.78	80.11
W/ ERD W/O SBAE	29 × 26	104/9	98.49	98.59	98.69	98.56
	27 × 25	37/8	99.12	99.25	99.13	99.25
	39 × 37	256/9	97.01	97.33	97.11	97.22
	All	76/9	92.32	92.44	92.11	92.11
W/O ERD W/ SBAE	29 × 26	169/9	97.83	98.11	98.33	98.33
	27 × 25	169/8	**99.48**	**99.63**	**99.38**	**99.5**
	39 × 37	169/9	92.80	94.11	92.89	92.89
	All	169/9	93.49	94.11	93.56	93.67
W/ ERD W/ SBAE	29 × 26	22/9	**98.85**	**98.89**	**98.89**	**99**
	27 × 25	19/8	99.24	99.25	99.25	99.38
	39 × 37	27/9	**97.37**	**97.89**	**97.22**	**97.33**
	All	45/9	**95.15**	**95.11**	**95.11**	**95**

Bold font is used to emphasize the best results

Table 12 Comparison of different common metaheuristic algorithms for the proposed fault detection in wafer maps

Wafer map resolution	Final feature size/classes	Metaheuristic algorithm	Acc	Pre	Sen	F1-score
25 × 27	168/8	GA	99.22	99.25	99.25	99.25
	216/8	EO	99.19	99.38	99.25	99.13
	162/8	SCA	99.31	99.38	99.25	99.25
	223/8	GWO	99.16	99.25	99.25	99.25
	174/8	PSO	99.19	99.25	99.25	99.25
	22/8	ERD	98.36	98.37	98.37	98.5
29 × 26	152/9	GA	99.53	99.67	99.56	99.67
	215/9	EO	99.58	99.67	99.67	99.67
	165/9	SCA	99.63	99.67	99.67	99.67
	167/9	GWO	99.61	99.67	99.67	99.56
	175/9	PSO	99.61	99.67	99.67	99.67
	22/9	ERD	98.85	98.89	98.89	99
30 × 34	164/9	GA	99.30	99.33	99.22	99.44
	219/9	EO	99.40	99.44	99.44	99.44
	181/9	SCA	99.19	99.22	99.22	99.22
	185/9	GWO	99.22	99.33	99.22	99.22
	175/9	PSO	99.37	99.44	99.44	99.44
	23/9	ERD	98.17	98.44	98.11	98.11
39 × 37	211/9	GA	98.42	98.78	98.56	98.44
	221/9	EO	98.27	98.67	98.22	98.44
	180/9	SCA	98.54	98.78	98.56	98.67
	197/9	GWO	98.04	98.33	98.11	98.11
	191/9	PSO	98.19	98.44	98.22	98.33
	27/9	ERD	97.37	97.89	97.22	97.33
33 × 29	164/9	GA	99.68	**99.78**	99.78	99.78
	217/9	EO	**99.41**	**99.44**	**99.44**	99.56
	153/9	SCA	**99.47**	**99.56**	**99.44**	99.44
	152/9	GWO	**99.39**	**99.44**	**99.44**	99.33
	166/9	PSO	**99.39**	**99.33**	**99.44**	99.44
	15/9	ERD	98.78	**98.89**	98.89	98.77

Bold font is used to emphasize the best results

Appendix

Tables 10, 11, 12, and 13 corresponds the Tables 6, 7, 8, and 9. They just show results at different other resolutions rather than those employed in Tables 6, 7, 8, and 9 in the main text. Table 10 indicates computitative comparison between the performance of different resolutions of wafer maps (26 × 26, 29 × 26, 30 × 34, and All) at different split ratios through the proposed inspection model. Table 11 demonstrates comparison between the different cases of applying the adopted feature engineering steps, i.e., ERD and SBAE, at resolutions (29 × 26, 27 × 25, 39 × 37, All). Table 12 introduces comparison of different common metaheuristic algorithms for the proposed fault detection in wafer maps, for resolutions (25 × 27, 29 × 26, 30 × 34, 39 × 37, and 33 × 29). Table 13 presents Comparison of different common 1D CNNs for the proposed fault detection in wafer maps, at resolutions (25 × 27, 29 × 26, 39 × 37, 33 × 29, and All). Table 10, 11, 12, and 13 resemble/correspond the set of Tables 6, 7, 8, and

Table 13 Comparison of different common 1D CNNs for the proposed fault detection in wafer maps

Wafer map resolution	Final feature size/classes	Classification technique	Acc	Pre	Sen	F1-score	No of parameters
25 × 27	22/8	1D-VGG16	98.38	98.5	98.5	98.5	594 K
		1D-ResNet50	96.36	96.5	96.38	96.38	16 M
		1D-LeNet-5	97.66	97.88	97.75	97.63	244 K
		1D-Inception	97.03	97	97.13	97.13	15 K
		Our 1DCNN	98.36	98.37	98.37	98.5	175 K
29 × 26	22/9	1D-VGG16	99.16	99.22	**99.22**	99.22	594 K
		1D-ResNet50	97.26	97.67	97.44	97.33	16 M
		1D-LeNet-5	98.56	98.78	98.67	98.56	244 K
		1D-Inception	97.73	97.78	97.78	97.78	16 K
		Our 1DCNN	**98.85**	98.89	98.89	**99**	176 K
39 × 37	27/9	1D-VGG16	96.86	97.67	96.78	96.78	626 K
		1D-ResNet50	96.34	97	96.22	97.22	16 M
		1D-LeNet-5	97.01	97.56	96.89	97	277 K
		1D-Inception	96.76	97.22	96.67	96.67	16 K
		Our 1DCNN	97.37	97.89	97.22	97.33	208 K
33 × 29	15/9	1D-VGG16	99.04	**99.11**	99	99.11	528 K
		1D-ResNet50	**97.01**	**97.11**	**97.11**	97.11	16 M
		1D-LeNet-5	**98.88**	**98.78**	**99**	98.89	178 K
		1D-Inception	**98.08**	**0.98**	**98.22**	98.33	16 K
		Our 1DCNN	98.78	**98.89**	98.89	98.77	176 K
All (26 × 26)	45/9	1D-VGG16	96.51	96.67	96.56	96.44	626 K
		1D-ResNet50	93.41	93.44	93.22	93.33	16 M
		1D-LeNet-5	96.21	96.22	96.11	96.11	277 K
		1D-Inception	92.95	93.22	92.89	92.78	16 K
		Our 1DCNN	95.15	95.11	95.11	95	176 K

Bold font is used to emphasize the best results

Table 14 Glossary of the main variables/functions/parameters in this paper

Variable	Annotation	Variable	Annotation
X	The original grey wafer map respectively of size (m, n)	$\mathcal{X}^s$	Are the resultant encoded sparse maps from SBAE
XX	The one-hot encoded colored wafer map of size (m, n, c)	y^s, Y^s	y^s of size of size $\left(1 \times \frac{mnc}{4}\right)$ is 1D sinogram of the previous 2D sparse map $\mathcal{X}^s$, so for N samples, we have sinogramic feature pool Y^s of size $\left(N \times \frac{mnc}{4}\right)$
$\mathcal{X}$	The synthesized wafer map after fixing the imbalance problem of size (m, n, c)	$G = [\mathcal{Y}_1, \mathcal{Y}_2, .., \mathcal{Y}_{\mathrm{N}_p}]$	Is a group of vertical features from Y^s. $Y^s = \left[\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_{\frac{mnc}{4}}\right]$. $Y^s = \left[y_1^s, y_2^s, \ldots, y_N^s\right]^T$. y_i^s is a sparse encoded sample. y_i is a sparse encoded feature vector
H	The convolutional output of a hidden layer	f	Is the employed fitness function in ERD algorithm
H_c	The convolutional output of a hidden layer after MaxPooling "down sampling" operation	u and ℓ	Are the upper and lower limits of local search of neighboring solutions in ERD
$W, B, \mathcal{A}$	The employed weights, bias, and activation function	$(\alpha_1, \alpha_2, \alpha_3, \delta)$; (β_1, β_2)	Are randomly generated coefficients from a uniform distribution that ranges from 0 to 1 in the roaring and fighting phases respectively
N	The total number of wafer maps after synthetization	N_i	Is the number of selected features under the category $i \in \{stag, com, hind, male\}$
$\mathcal{C}(W, B)$	Is the cost function of Sparsity-boosted autoencoder (SBAE)	G_i	Is the grouped features under the category $i \in \{stag, com, hind, OS, tink\}$
$\mathcal{K}(\mathcal{P} \Vert \widehat{\mathcal{P}})$	is Kullback–Leibler divergence	v	Is the diversity in fitting between the tinkered harem and the tinkering harem and its threshold value ζ is used to check how the new offspring RDs, G_{OS}, are generated
$\mathcal{P}, \widehat{\mathcal{P}}$	$\mathcal{P}$ is the targeted or assigned sparsity, while $\widehat{\mathcal{P}}$ is average output of all hidden neurons (the actual resultant sparsity amount)	$\hat{\mathrm{n}}$	Is a fixed number of harems to be updated within each iteration
$\mathcal{K}, \gamma$	Are weighting coefficients in $\mathcal{C}(W, B)$	θ	Is a random number between 0 and 1 in the mating phase

9. Table 10, 11, 12, and 13 just shows results at different resolutions rather than those mentioned in the main text in Tables 6, 7, 8, and 9. Table 14 indicates summary of the main variables/parameters/functions mentioned in this paper.

Author contributions All persons who meet authorship criteria are listed as authors, and all authors certify that they have participated sufficiently in the work to take public responsibility for the content, including participation in the concept, design, analysis, writing, or revision of the manuscript.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availability All employed datasets are open source and have been cited in this paper.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Metaheuristic algorithms: A comprehensive review. In *Computational intelligence for multimedia big data on the cloud with engineering applications* (pp. 185–231). https://doi.org/10.1016/b978-0-12-813314-9.00010-4

Alam, L., & Kehtarnavaz, N. (2022). A survey of detection methods for die attachment and wire bonding defects in integrated circuit manufacturing. *IEEE Access, 10*, 83826–83840. https://doi.org/10.1109/access.2022.3197624

Alqudah, R., Al-Mousa, A. A., Hashyeh, Y. A., & Alzaibaq, O. Z. (2023). A systemic comparison between using augmented data and synthetic data as means of enhancing wafer map defect classification. *Computers in Industry, 145*, 103809. https://doi.org/10.1016/j.compind.2022.103809

Altantawy, D. A., & Kishk, S. S. (2023). Equilibrium-based COVID-19 diagnosis from routine blood tests: A sparse deep convolutional model. *Expert Systems with Applications, 213*, 118935. https://doi.org/10.1016/j.eswa.2022.118935

Baly, R., & Hajj, H. (2012). Wafer classification using support vector machines. *IEEE Transactions on Semiconductor Manufacturing, 25*(3), 373–383. https://doi.org/10.1109/tsm.2012.2196058

Chen, S., Zhang, Y., Hou, X., Shang, Y., & Yang, P. (2022). Wafer map failure pattern recognition based on deep convolutional neural network. *Expert Systems with Applications, 209*, 118254. https://doi.org/10.1016/j.eswa.2022.118254

Chen, S., Zhang, Y., Yi, M., Shang, Y., & Yang, P. (2021). AI classification of wafer map defect patterns by using dual-channel convolutional neural network. *Engineering Failure Analysis, 130*, 105756. https://doi.org/10.1016/j.engfailanal.2021.105756

Cheng, K. C. C., Chen, L. L. Y., Li, J. W., Li, K. S. M., Tsai, N. C. Y., Wang, S. J., Huang, A. Y. A., Chou, L., Lee, C. S., Chen, J. E., & Liang, H. C. (2021). Machine learning-based detection method for wafer test induced defects. *IEEE Transactions on Semiconductor Manufacturing, 34*(2), 161–167. https://doi.org/10.1109/tsm.2021.3065405

Chu, M., Park, S., Jeong, J., Joo, K., Lee, Y., & Kang, J. (2022). Recognition of unknown wafer defect via optimal bin embedding technique. *The International Journal of Advanced Manufacturing Technology, 121*(5–6), 3439–3451. https://doi.org/10.1007/s00170-022-09447-y

Faris, H., Aljarah, I., Al-Betar, M. A., & Mirjalili, S. (2018). Grey wolf optimizer: A review of recent variants and applications. *Neural Computing and Applications, 30*, 413–435. https://doi.org/10.1007/s00521-017-3272-5

Fathollahi-Fard, A. M., Hajiaghaei-Keshteli, M., & Tavakkoli-Moghaddam, R. (2020). Red deer algorithm (RDA): A new nature-inspired meta-heuristic. *Soft Computing, 24*, 14637–14665. https://doi.org/10.1007/s00500-020-04812-z

Houssein, E. H., Çelik, E., Mahdy, M. A., & Ghoniem, R. M. (2022). Self-adaptive Equilibrium Optimizer for solving global, combinatorial, engineering, and Multi-Objective problems. *Expert Systems with Applications, 195*, 116552. https://doi.org/10.1016/j.eswa.2022.116552

Jin, C. H., Kim, H. J., Piao, Y., Li, M., & Piao, M. (2020). Wafer map defect pattern classification based on convolutional neural network features and error-correcting output codes. *Journal of Intelligent Manufacturing, 31*(8), 1861–1875. https://doi.org/10.1007/s10845-020-01540-x

Jin, C. H., Na, H. J., Piao, M., Pok, G., & Ryu, K. H. (2019). A novel DBSCAN-based defect pattern detection and classification framework for wafer bin map. *IEEE Transactions on Semiconductor Manufacturing, 32*(3), 286–292. https://doi.org/10.1109/tsm.2019.2916835

Kang, H., & Kang, S. (2021). A stacking ensemble classifier with handcrafted and convolutional features for wafer map pattern classification. *Computers in Industry, 129*, 103450. https://doi.org/10.1016/j.compind.2021.103450

Kim, D., & Kang, P. (2021). Dynamic clustering for wafer map patterns using self-supervised learning on convolutional autoencoders. *IEEE Transactions on Semiconductor Manufacturing, 34*(4), 444–454. https://doi.org/10.1109/tsm.2021.3107720

Kim, T., & Behdinan, K. (2023). Advances in machine learning and deep learning applications towards wafer map defect recognition and classification: A review. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-022-01994-1

Kim, T. S., Lee, J. W., Lee, W. K., & Sohn, S. Y. (2021). Novel method for detection of mixed-type defect patterns in wafer maps based on a single shot detector algorithm. *Journal of Intelligent Manufacturing*. https://doi.org/10.1007/s10845-021-01755-6

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1D convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing, 151*, 107398. https://doi.org/10.1016/j.ymssp.2020.107398

Kyeong, K., & Kim, H. (2018). Classification of mixed-type defect patterns in wafer bin maps using convolutional neural networks. *IEEE Transactions on Semiconductor Manufacturing, 31*(3), 395–402. https://doi.org/10.1109/tsm.2018.2841416

Leavers, V. F. (1992). Use of the Radon transform as a method of extracting information about shape in two dimensions. *Image and Vision Computing, 10*(2), 99–107. https://doi.org/10.1016/0262-8856(92)90004-m

Lee, S. H., Koo, H. I., & Cho, N. I. (2010). New automatic defect classification algorithm based on a classification-after-segmentation framework. *Journal of Electronic Imaging, 19*(2), 020502. https://doi.org/10.1117/1.3429116

Li, P., Pei, Y., & Li, J. (2023). A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*. https://doi.org/10.1016/j.asoc.2023.110176

Liu, C. W., & Chien, C. F. (2013). An intelligent system for wafer bin map defect diagnosis: An empirical study for semiconductor manufacturing. *Engineering Applications of Artificial Intelligence, 26*(5–6), 1479–1486. https://doi.org/10.1016/j.engappai.2012.11.009

Nag, S., Makwana, D., Mittal, S., & Mohan, C. K. (2022). WaferSegClassNet—A light-weight network for classification and segmentation of semiconductor wafer defects. *Computers in Industry, 142*, 103720. https://doi.org/10.1016/j.compind.2022.103720

Ng, A. (2011). Sparse autoencoder. CS294A Lecture Notes, *72*, 1–19. https://graphics.stanford.edu/courses/cs233-21-spring/ReferencedPapers/SAE.pdf

Saqlain, M., Jargalsaikhan, B., & Lee, J. Y. (2019). A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing, 32*(2), 171–182. https://doi.org/10.1109/tsm.2019.2904306

Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. *IEEE Access, 10*, 10031–10061. https://doi.org/10.1109/access.2022.3142859

Shankar, N. G., & Zhong, Z. W. (2005). Defect detection on semiconductor wafer surfaces. *Microelectronic Engineering, 77*(3–4), 337–346. https://doi.org/10.1016/j.mee.2004.12.003

Shin, W., Kahng, H., & Kim, S. B. (2022). Mixup-based classification of mixed-type defect patterns in wafer bin maps. *Computers and Industrial Engineering, 167*, 107996. https://doi.org/10.1016/j.cie.2022.107996

Sohail, A. (2023). Genetic algorithms in the fields of artificial intelligence and data sciences. *Annals of Data Science, 10*(4), 1007–1018. https://doi.org/10.1007/s40745-021-00354-9

Sun, Y., Song, Q., & Liang, F. (2022). Consistent sparse deep learning: Theory and computation. *Journal of the American Statistical Association, 117*(540), 1981–1995. https://doi.org/10.1080/01621459.2021.1895175

Theodosiou, T., Rapti, A., Papageorgiou, K., Tziolas, T., Papageorgiou, E., Dimitriou, N., Margetis, G., & Tzovaras, D. (2023). A review study on ML-based methods for defect-pattern recognition in wafer maps. *Procedia Computer Science, 217*, 570–583. https://doi.org/10.1016/j.procs.2022.12.253

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(11). http://jmlr.org/papers/v9/vandermaaten08a.html

Wang, F. K., Chou, J. H., & Amogne, Z. E. (2022). A deep convolutional neural network with residual blocks for wafer map defect pattern recognition. *Quality and Reliability Engineering International, 38*(1), 343–357. https://doi.org/10.1002/qre.2983

Wang, S., Zhong, Z., Zhao, Y., & Zuo, L. (2021). A variational autoencoder enhanced deep learning model for wafer defect imbalanced classification. *IEEE Transactions on Components, Packaging and Manufacturing Technology, 11*(12), 2055–2060. https://doi.org/10.1109/tcpmt.2021.3126083

WM-811K (2014). Retrieved March 15, 2014, from https://www.kaggle.com/datasets/qingyi/wm811k-wafer-map

Wu, M. J., Jang, J. S. R., & Chen, J. L. (2014). Wafer map failure pattern recognition and similarity ranking for large-scale data sets. *IEEE Transactions on Semiconductor Manufacturing, 28*(1), 1–12. https://doi.org/10.1109/tsm.2014.2364237

Xu, Q., Yu, N., & Hasan, M. M. (2023). Evolutionary computation-based reliability quantification and its application in big data analysis on semiconductor manufacturing. *Applied Soft Computing, 136*, 110080. https://doi.org/10.1016/j.asoc.2023.110080

Xuen, L. S., Mohd Khairuddin, I., Mohd Razman, M. A., Mat Jizat, J. A., Yuen, E., Jiang, H., Yap, E. H., & Abdul Majeed, P. P. A. (2022, December). The classification of wafer defects: A support vector machine with different DenseNet transfer learning models evaluation. In *International conference on robot intelligence technology and applications* (pp. 304–309). Springer. https://doi.org/10.1007/978-3-031-26889-2_27

Yan, J., Sheng, Y., & Piao, M. (2023). Semantic segmentation based wafer map mixed-type defect pattern recognition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. https://doi.org/10.1109/tcad.2023.3274958

Yoon, S., & Kang, S. (2022). Semi-automatic wafer map pattern classification with convolutional neural networks. *Computers and Industrial Engineering, 166*, 107977. https://doi.org/10.1016/j.cie.2022.107977

Yu, J., Shen, Z., & Wang, S. (2021b). Wafer map defect recognition based on deep transfer learning-based densely connected convolutional network and deep forest. *Engineering Applications of Artificial Intelligence, 105*, 104387. https://doi.org/10.1016/j.engappai.2021.104387

Yu, J., Li, S., Shen, Z., Wang, S., Liu, C., & Li, Q. (2021a). Deep transfer Wasserstein adversarial network for wafer map defect recognition. *Computers and Industrial Engineering, 161*, 107679. https://doi.org/10.1016/j.cie.2021.107679

Yu, J., Zheng, X., & Liu, J. (2019). Stacked convolutional sparse denoising auto-encoder for identification of defect patterns in semiconductor wafer map. *Computers in Industry, 109*, 121–133. https://doi.org/10.1016/j.compind.2019.04.015

Yu, N., Chen, H., Xu, Q., Hasan, M. M., & Sie, O. (2022). Wafer map defect patterns classification based on a lightweight network and data augmentation. *CAAI Transactions on Intelligence Technology*. https://doi.org/10.1049/cit2.12126

Yu, N. G., Xu, Q., Wang, H. L., & Lin, J. (2021c). Wafer bin map inspection based on DenseNet. *Journal of Central South University, 28*(8), 2436–2450. https://doi.org/10.1007/s11771-021-4778-7

Zhang, Q., Zhang, Y., Li, J., & Li, Y. (2022). WDP-BNN: Efficient wafer defect pattern classification via binarized neural network. *Integration, 85*, 76–86. https://doi.org/10.1016/j.vlsi.2022.04.003

Zheng, H., Sherazi, S. W. A., Son, S. H., & Lee, J. Y. (2021). A deep convolutional neural network-based multi-class image classification for automatic wafer map failure recognition in semiconductor manufacturing. *Applied Sciences, 11*(20), 9769. https://doi.org/10.3390/app11209769

Zhou, W., Wang, P., Heidari, A. A., Zhao, X., & Chen, H. (2022). Spiral Gaussian mutation sine cosine algorithm: Framework and comprehensive performance optimization. *Expert Systems with Applications, 209*, 118372. https://doi.org/10.1016/j.eswa.2022.118372