Computers & Industrial Engineering 172 (2022) 108614

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

An ensemble-based deep semi-supervised learning for the classification of Wafer Bin Maps defect patterns

Siyamalan Manivannan

Department of Computer Science, Faculty of Science, University of Jaffna, Sri Lanka

ARTICLE INFO

Keywords:
Semiconductor manufacturing
Semi-supervised learning
Convolutional Neural Networks

ABSTRACT

Wafer is a thin slice of semiconductor substance used for fabricating integrated circuits in semiconductor manufacturing. Wafer Bin Maps (WBM) are the results of Circuit Probe inspection of the dices on the wafer, which provide crucial information to identify the root cause of the problems in semiconductor manufacturing. Automatic identification of defect patterns in WBMs remains a challenging problem due to the availability of the labeled data. Deep Convolutional Neural Networks (CNN) based fully supervised approaches have already been investigated and satisfactory classification performance have been obtained for the classification of WBM defect patterns. However, as they are fully supervised approaches, they require labeled data for training. Obtaining large amount of labeled data is a tedious and time consuming process. To overcome this, in this work we propose a CNN ensemble based semi-supervised approach, which make use of both labeled and unlabeled data for training. One of the main problem with CNN is that they often produce high-confident predictions, even for wrongly classified samples. We overcome this problem by the use of both *Label-Smoothing* and *Ensembling*. Comparative experiments on a large scale, public WBM dataset, *WM-811K* show that the proposed method is the new state-of-the-art, and we show that our approach outperforms other approaches even with relatively low amount of labeled data used for training.

1. Introduction

Semiconductor manufacturing is one of the complex, lengthy and costly manufacturing process, in which with the aid of advanced tools a variety of Integrated Circuits including microprocessors, memories, digital signal processors, etc. are fabricated on thin slices of semiconductor substances called wafers. In order to produce high quality integrated circuits wafers must be clean and defect free. Therefore, Circuit Probe test is performed on each of the dices of the fabricated wafer to detect specific failures. WBM is the result of this test for a particular wafer, which indicates whether each of the dice on this wafer is a defective one or non defective one, presenting spatial defective patterns on the wafer (Fig. 1). By analyzing these spatial patterns corresponding process failures can be identified. For example, in 'center' pattern type, most defective dies are in the center of the wafer map, and this pattern is the cause of uniformity problem during the chemical–mechanical planarization (Jin, Kim, Piao, Li, & Piao, 2020). Inappropriate wafer handling or problems with shipment can cause the 'scratch' patterns, and a layer-to-layer misalignment during storage can cause the 'edge-ring' patterns (Jin et al., 2020).

Most companies rely on experienced engineers for the visual inspection of the WBM images to analyze the spatial patterns appear on them to identify the corresponding cause (Liu & Chien, 2013). This manual process is time consuming, expensive, unreliable, subjective, and requires experience and expert knowledge. To overcome this, vision based automatic inspection methods are proposed, as they are fast, highly accurate and significantly reduce labor intensity. However, automatic identification of defects is a challenging task due to limited labeled training data, and high intra and low inter-class variations of the WBM patterns.

The majority of the approaches proposed in the literature for WBM image classification (e.g. Adly, Alhussein et al., 2015; Adly, Yoo et al., 2015; Chien, Hsu, & Chen, 2013; Jeong, Kim, & Jeong, 2008; Li & Huang, 2009; Piao, Jin, Lee, & Byun, 2018; Wu, Jang, & Chen, 2015) are supervised learning based approaches, which require large quantities of labeled data for training. Obtaining labeled data in such quantities is a tedious, time consuming, expensive process, and requires expert knowledge. To overcome this, semi-supervised learning approaches become popular in the computer vision community (e.g., Arazo, Ortego, Albert, O'Connor, & McGuinness, 2020; Van Engelen & Hoos, 2020; Xie, Luong, Hovy, & Le, 2020; Yang, Song, King, & Xu, 2021) as they have the ability to learn from both labeled and unlabeled data. However, to best of our knowledge these semi-supervised approaches have not been well explored for the classification of WBM.

E-mail address: siyam@univ.jfn.ac.lk.

https://doi.org/10.1016/j.cie.2022.108614
Received 4 April 2022; Received in revised form 21 July 2022; Accepted 24 August 2022
Available online 30 August 2022

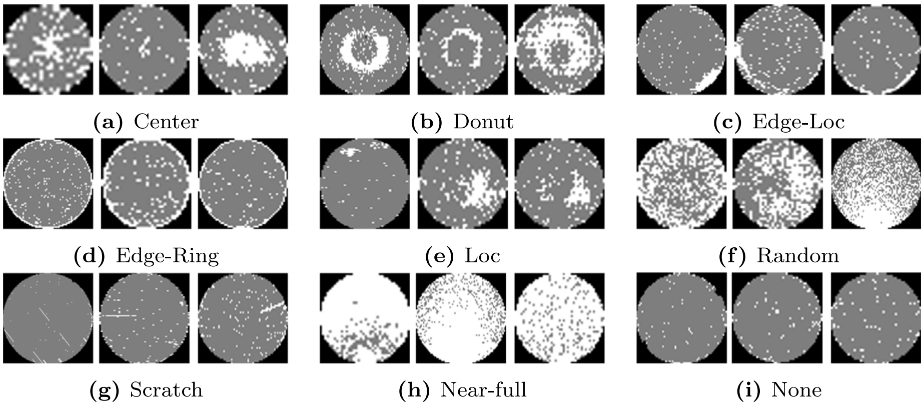

Fig. 1. Example images from different spatial defect pattern categories of the WM-811K dataset [1].

In this work, we propose an ensemble based semi-supervised deep learning approach for WBM image classification. In our proposed approach, initially, a set of CNN models are trained independently from each other using the labeled dataset. Then, this ensemble classifier is used to predict the labels (*pseudo-labels*, i.e., the target labels) of each unlabeled image in the unlabeled dataset. In addition, each of this unlabeled image is also weighted based on how well it is classified by the ensemble classifier. These weighted samples and their corresponding pseudo-labels are then used to update each CNN in this ensemble. At test time, the ensemble classifier is used to predict the label of any given image. However, CNN often produces over-confident predictions, even for wrongly classified images (Nguyen, Yosinski, & Clune, 2015), which make it hard to identify the correct pseudo-label of each unlabeled image. We overcome this problem by the use of both *Label Smoothing* (Nguyen et al., 2015) and *Ensembling*, and show this combination is the key to success of the proposed method.
The main contributions of this work include:

- An ensembling and label smoothing based semi-supervised deep learning approach for the classification of WBM images.
- The use of the combination of label smoothing and ensembling techniques to overcome the problems associated with overconfident predictions when finding the pseudo-labels of the unlabeled images for training.
- The use of soft-weighting over hard-weighting for weighting the contribution of each unlabeled sample based on how well that sample is predicted.

We experimentally show that our system beats the other state-of-the-art systems with relatively low amount of labeled data used for training.

The rest of the paper is organized as follows: Section 2 summarizes the related work. The proposed methodology is explained in detail in Section 3. The dataset, experiments, and the results with discussion are reported in Section 4. Finally, Section 5 concludes this paper.

2. Related work

The approaches proposed for WBM image classification can be categorized based on the type of labeling used, into: unsupervised, supervised and semi-supervised approaches. This section summarizes these approaches under these categories.

2.1. Unsupervised approaches

These approaches (e.g. Chang, Li, Chang, and Jeng (2009), Hsu (2015), Liu and Chien (2013), Liukkonen and Hiltunen (2018), Palma, Nicolao, Miraglia, Pasquinetti, and Piccinini (2005), Tulala, Mahyar, Ghalebi, and Grosu (2018), Wang, Kuo, and Bensmail (2006)) mainly based on clustering techniques for the identification of defect types. For example, Wang et al. (2006) proposed a hybrid approach, where a spatial filter is applied to judge whether the input data contains any systematic clusters, and then a K-Means clustering and a spherical-shell algorithm were used for the identification of different defect patterns. Liu and Chien (2013) proposed different clustering approaches based on spatial statistics. Tulala et al. (2018) proposed an Auto Encoder based approach, where an Auto Encoder is used to project WBM images into a low-dimensional latent representation, and then K-Means clustering is applied to find the clusters using these representations. Self-organized map based approaches (Chang et al., 2009; Liukkonen & Hiltunen, 2018; Palma et al., 2005) were also explored for WBM clustering. Although unsupervised approaches do not require labeled data for training, they may still require some expert knowledge to identify which clusters correspond to which defect patterns.

2.2. Supervised approaches

Supervised approaches usually give superior classification performance than unsupervised approaches. The supervised approaches proposed for WBM classification can be categorized into (1) shallow learning based approaches, and (2) deep learning based approaches.

The shallow learning based approaches rely on hand-crafted features and machine learning based classifiers. Various features such as rotation and scale invariant features (Wu et al., 2015), Radon transform based features (Piao et al., 2018), spatial autocorrelations based features (Jeong et al., 2008), co-occurrence matrix based features (Li & Huang, 2009), morphological based features (Liao, Hsieh, Huang, & Chien, 2014), spatial statistical features (Chien et al., 2013), and classifiers such as SVM (Li & Huang, 2009; Wu et al., 2015), Decision Tree ensembles (Piao et al., 2018), neural networks (Chien et al., 2013), regression network based classifiers (Adly, Alhussein et al., 2015; Adly, Yoo et al., 2015) were explored for WBM image classification.

Usually shallow learning approaches give inferior performance compared to deep learning approaches, as in shallow learning the features are often hand-crafted, and therefore may not capture discriminative information. On the other hand, deep learning approaches provide state-of-the-art results for various tasks including WBM image classification as they have the ability to train a system in an end-to-end manner, and therefore can capture most discriminative features and classifier. A significant amount of deep learning based approaches has been proposed recently for WBM image classification. Most of these approaches focus on applying different CNN architectures for this problem; MobileNet (Tsai & Lee, 2020), LeNet (Hsu & Chien, 2020), AlexNet (Hsu & Chien, 2020), GoogleNet (Hsu & Chien, 2020) and Region Proposal networks (Chien, Wu, & Lee, 2020) are among

[1] https://www.kaggle.com/qingyi/wm811k-wafer-map

a few. Wang and Chen (2020) trained a custom CNN on the polar mapped transformed images. Nakazawa and Kulkarni (2018) proposed a method for synthetic data generation using Poisson process, and trained a CNN on these synthetically generated images. Hsu and Chien (2020) proposed an ensemble CNN, where a set of CNN were trained and a weighted majority of their predictions were considered as the final prediction for each WBM image. Jin et al. (2020) used the features extracted from a CNN to train a SVM classifier. Batool, Shapiai, Fauzi, and Fong (2020) proposed to use under-sampling technique to handle imbalanced data when training a CNN.

2.3. *Semi-supervised approaches*

Although supervised approaches provide state-of-the-art results for various tasks, they usually require large amount of labeled data for training. Obtaining labeled data in large quantities is time and cost consuming process, and requires expert knowledge. To overcome this, semi-supervised approaches becoming popular as they have the ability to learn from both labeled and unlabeled data. However, to best of our knowledge, only a few attempts (Kahng & Kim, 2021; Kong & Ni, 2018, 2020) based on semi-supervised learning were explored for the classification of WBM images. For example, Kahng and Kim (2021) recently proposed a self-representation learning framework, where first the unlabeled images were used to pre-train a CNN based on minimizing the distance between the features extracted from the original WBM images and their augmented versions. After this pre-training stage, the CNN is fine-tuned with the labeled data for classification, and showed improved classification performance compared to only using the labeled data for training the CNN. A semi-supervised approach based on a CNN and an Auto Encoder was proposed in Kong and Ni (2018, 2020), where the auto encoder was trained in an unsupervised manner by minimizing the differences between the outputs obtained from the clean and their corresponding corrupted WBM samples. The CNN classifier is then trained in a supervised manner on top of the features extracted from the auto encoder.

2.4. *Summary*

As a summary, although there were various supervised approaches proposed for WBM image classification, semi-supervised learning approaches were not well explored. Therefore, in this work, we propose a semi-supervised deep learning approach, which uses an ensembling and label smoothing technique for training the CNN in an end-to-end manner with both the labeled and unlabeled data, and show state-of-the-art results on a large scale WBM image dataset.

3. Methodology

In this section we propose an ensemble based semi-supervised deep learning approach for the classification of WBM. In summary, first a set of CNN models are trained in a fully supervised manner using the available labeled data for a particular number of epochs. Then the unlabeled data is added with the labeled set for semi-supervised training. At each iteration of the semi-supervised training, the pseudo-label of each unlabeled sample is calculated using this ensemble classifier. In addition, the weight of each sample is also calculated based on how well it is predicted using this ensemble classifier. These weighted pseudo-labeled samples are then used to update each of the CNN classifier in the ensemble. This semi-supervised training is continued for a particular number of epochs. At the test time, this ensemble classifier is used for prediction. In the following, we first explain the notations that we use, and then explain the proposed methodology in detail.

3.1. *Notations*

All the notations used in this paper are summarized in Table 1.

The main aim of this work is to train an ensemble of CNN models from both labeled ($\mathcal{D}_l$) and unlabeled ($\mathcal{D}_u$) data. Let us assume that $\mathcal{D}_l$ contains a set of tuples $\{I_i, \mathbf{y}_i\}$, where, I_i represents the ith image, and $\mathbf{y}_i$ its label in one-hot representation respectively, and there are C number of classes. On the other hand, $\mathcal{D}_u$ contains a set of images, and their labels are unknown.

Let $\mathcal{F}_\theta$ represents a CNN, where θ are its parameters which need to be learned. In a C class classification problem, the last layer of the CNN will have C nodes. When the ith image, I_i, is passed through this CNN the network will provide C output values, i.e,

$$\mathbf{z}_i = \mathcal{F}_\theta(I_i) \tag{1}$$

where, $\mathbf{z}_i \in \mathbb{R}^C$ is a vector which contains the C output predictions for I_i.

Now, if K such networks are considered in an ensemble, the predicted vector from the kth network for the image I_i can be given as (Fig. 2):

$$\mathbf{z}_i^k = \mathcal{F}_\theta^k(I_i) \tag{2}$$

The classification probability of the image I_i belonging to class c calculated from the kth CNN, p_{ic}^k, can be obtained by applying a softmax function on $\mathbf{z}_i^k$ as follows:

$$p_{ic}^k = \frac{e^{z_{ic}^k}}{\sum_{m=1}^C e^{z_{im}^k}} \tag{3}$$

where, z_{im}^k is the mth element of $\mathbf{z}_i^k$.

3.2. *Handling high-confident predictions*

Although CNNs have proven to be very good classifiers, they often overfit to the training data and produce high-confident predictions even for wrongly classified samples (Nguyen et al., 2015). Since our approach is based on pseudo-labels, the selected pseudo-labels for training must be correct as possible. Otherwise a large amount of wrongly selected pseudo-labels can easily leads to noisy training, and therefore, can produce worse classification results. This is the case with WBM (refer Section 4.5 for supporting experiments). To overcome this we use two different approaches: Label smoothing and Ensembling.

3.2.1. *Label smoothing*

When hard-labels (i.e., each image is assigned to only one class) are used for training a CNN with cross entropy loss, the trained CNN model may result in overfitting as for each training sample the model learns to assign full probability to the class which corresponds to the groundtruth (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016). In addition, the model will encourage the differences between the largest probability and all others to become large, and therefore, even the wrongly classified samples will have high confident predictions (refer Table 6 and Section 4.5 for supporting experiments). To overcome these issues, Label Smoothing was introduced in Szegedy et al. (2016) as a way of regularizing deep learning models, and has been widely used in many state-of-the-art image classification problems.

In label smoothing, the hard-labels ($\mathbf{y}_i$) are converted into soft-labels ($\mathbf{y}_i^s$) by the introduction of a smoothing factor α, and then the deep learning model is trained on these soft-labels. The soft-labels obtained in this manner can be defined as:

$$\mathbf{y}_i^s = (1-\alpha)\mathbf{y}_i + \frac{\alpha}{C} \tag{4}$$

For example, let us consider a five-class image classification problem, where, an image I_i corresponds to class 2, therefore, its hard-label (one-hot representation) can be given as: $\mathbf{y}_i = [0, 1, 0, 0, 0]$. When the

Table 1
Notations used throughout this paper.

Notation	Description
$\mathcal{D}_l$	labeled training set
$\mathcal{D}_u$	unlabeled training set
C	number of classes
I_i	an image
$\mathbf{y}_i$, y_{ic}	one-hot representation of the label of I_i and y_{ic} is an element of $\mathbf{y}_i$
$\mathbf{y}_i^s$	soft-label of y_i (after applying label smoothing)
$\hat{\mathbf{y}}_i$	one-hot representation of the pseudo-label of I_i calculated from the ensemble classifier
$\hat{\mathbf{y}}_i^s$	label smoothed version of $\hat{\mathbf{y}}_i$
α	the smoothing factor for label smoothing
$\mathcal{F}_\theta^k$	k^{th} CNN classifier in the ensemble
K	number of CNN classifiers in the ensemble
$\mathbf{z}_i^k$	A vector of size C which contains the predicted values of I_i by $\mathcal{F}_\theta^k$.
z_{ic}^k	The c^{th} element of $\mathbf{z}_i^k$
p_{ic}^k	probability of I_i belonging to class c obtained by $\mathcal{F}_\theta^k$. $\mathbf{p}_i^k = \left[p_{i1}^k, \dots p_{iC}^k\right]$
q_{ic}	probability of I_i belonging to class c obtained by the ensemble classifier

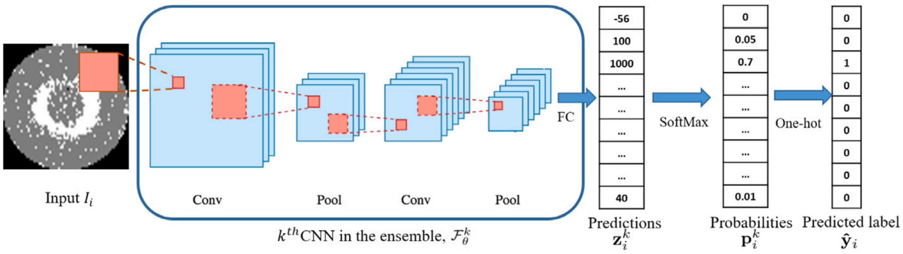

Fig. 2. An example CNN which shows some notations used throughout this manuscript (FC — Fully Connected layer).

smoothing factor α is set to $\alpha = 0.1$, its soft-label becomes: $\mathbf{y}_i^s = [0.02, 0.92, 0.02, 0.02, 0.02]$.

These soft-labels prevent the largest probability of each sample becoming larger and larger as all the classes of a particular image have non-zero contribution to the loss function, and therefore, prevents the model both from overfitting and from providing over confident predictions.

3.2.2. Ensembling

If the pseudo-labels are calculated based on only one CNN classifier, they may represent wrong classes, and therefore, subsequent training of the CNN using these pseudo-labels may lead to noisy training. To overcome this, we use an ensemble classifier, and calculate the pseudo-labels using this ensemble classifier. The predicted probability based on this ensemble for the image I_i belonging to class c can be given as:

$$q_{ic} = \frac{\bar{p}_{ic}}{\sum_{l=1}^{C} \bar{p}_{il}} \tag{5}$$

where, $\bar{p}_{ic}$ is the average of the probability values obtained from all the classifiers in the ensemble, and can be given as:

$$\bar{p}_{ic} = \frac{1}{K} \sum_{k=1}^{K} p_{ic}^k \tag{6}$$

The one-hot representation of the pseudo-label $\hat{\mathbf{y}}_i$ of an unlabeled image can be obtained as the image which gives the maximum prediction, where the cth element of $\hat{\mathbf{y}}_i$ can be given as:

$$\hat{y}_{ic} = \begin{cases} 1, & \text{if } c = \operatorname{argmax}_{c'} \{q_{ic'}\} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

The label smoothed version of this pseudo-label, $\hat{\mathbf{y}}_i^s$, can be obtained from Eq. (4).

3.3. Loss function

We use the following loss function for training the kth CNN of the ensemble with the training set which consists of both the labeled and unlabeled images.

$$\mathcal{L}^k = -\frac{1}{\mathcal{N}} \sum_{c=1}^{C} w_c \left[\sum_{i \in \mathcal{D}_l} y_{ic}^s \log p_{ic}^k + \sum_{i \in \mathcal{D}_u} u_i \, \hat{y}_{ic}^s \log p_{ic}^k \right] \tag{8}$$

This loss function can be decomposed into two terms: the first term represents the standard cross-entropy loss based on the labeled training data, and on the other hand, the second term represents the cross-entropy loss of the pseudo-labeled images from the unlabeled training set. Here, y_{ic}^s represents the label-smoothed version of the training label of the image I_i belonging to class c, and $\hat{y}_{ic}^s$ is the label-smoothed version of the pseudo-label which is obtained based on the ensemble classifier for the image I_i belonging to class c, as explained in Sections 3.2.1 and 3.2.2.

In this loss function, u_i is the weight applied to the unlabeled image I_i, which indicates how much one can trust the pseudo-labeling of that image. u_i gets a high value ($u_i \to 1$), if I_i is predicted with high confidence, i.e., $q_{ic} \to 1$, and low value, otherwise, as determined as follows:

$$u_i = \frac{1}{1 + \exp^{-\beta(q_{ic} - \tau)}} \tag{9}$$

where, q_{ic} is the probability obtained from the ensemble classifier as explained in Section 3.2.2. β and τ are tunable parameters which determine the *softness* of the weights. In the experiments, these parameters were set to $\beta = 30$ and $\tau = 0.9$ respectively. Section 4.6 investigates how these parameters affect the classification performance and compare this soft-weighting scheme with a hard-weighting scheme, and show improved performance by soft-weighting over hard-weighting.

In Eq. (8), w_c is the weight applied to class c to handle imbalanced number of images in different classes, and are determined as the inverse frequency of images in each class. As the number of images (with $u_i > 0$) in different classes in each mini-batch is changing overtime due to semi-supervised training, we calculate the class weights, w_c, for each mini-batch based on the following equation:

$$w_c = \frac{1}{\sum_{i \in \mathcal{D}_l^b} \mathbb{1}_{y_{ic}=1} + \sum_{i \in \mathcal{D}_u^b} \mathbb{1}_{q_{ic} > \tau}} \tag{10}$$

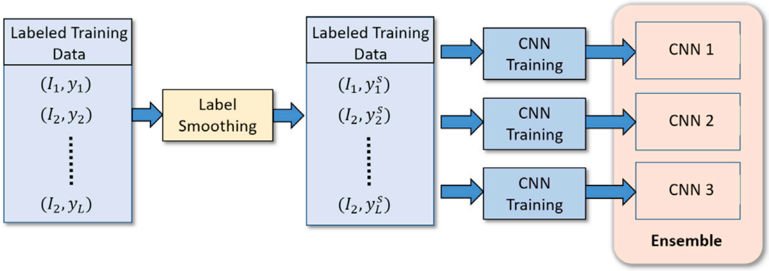

Fig. 3. Supervised pre-training.

Algorithm 1: An ensemble-based deep semi-supervised learning.

```
Input: Labeled dataset $\mathcal{D}_l$, Unlabeled dataset $\mathcal{D}_u$, size of the
       ensemble $K$, label smoothing factor $\alpha$, parameters to
       calculate instance weights $\beta$ and $\tau$, $nepochs$ - number of
       epochs for semi-supervised training
Output: An ensemble classifier $\{\mathcal{F}_\theta^k\}_{k=1}^K$, where $\mathcal{F}_\theta^k$ is the $k^{\text{th}}$
        CNN in the ensemble
// Supervised training
for k=1 to K do
    Train $\mathcal{F}_\theta^k$ using $\mathcal{D}_l$
end
// Semi-supervised training
for epoch in nepochs do
    foreach mini-batch $b$ in $\mathcal{D}_l \cup \mathcal{D}_u$ do
        foreach unlabeled sample $x$ in $b$ do
            Calculate the pseudo-label of $x$ based on the
             ensemble $\{\mathcal{F}_\theta^k\}_{k=1}^K$ using Eq. (7)
            Calculate the instance weight of $x$ using Eq. (9)
        end
        Calculate the class weights using Eq. (10)
        Apply label smoothing both on the original labels of the
         labeled, and pseudo-labels of the unlabeled samples
         using Eq. (4)
        for k=1 to K do
            Calculate the loss using Eq. (8) and update $\mathcal{F}_\theta^k$
        end
    end
end
```

where, $\mathbb{1}_x$ is the indicator function, returns 1 if x is true, and 0 otherwise. $\mathcal{D}_l^b$ and $\mathcal{D}_u^b$ represent the labeled and the unlabeled set of images that are sampled in a particular mini-batch b. This weight basically represents the inverse frequency of all the images from a particular class which are selected from both the labeled and the unlabeled datasets in a particular batch. But, here, not all the images are selected from the unlabeled set, instead, only the images with high-confident predictions are selected.

$\mathcal{N}$ in Eq. (8) is a normalization factor, and usually it is set as the size of the training set. In semi-supervised learning usually the training set contains a large number of unlabeled images than labeled images. As in each batch the images are sampled randomly from the training set which consists of both labeled and unlabeled images, sometimes a particular mini-batch may not contain any labeled images, and the number of unlabeled images with weights greater than zero (i.e., $u_i > 0$) also varies. To handle this, the normalization factor $\mathcal{N}$ is calculated for each mini-batch as follows:

$$\mathcal{N} = \sum_{c=1}^{C} w_c \left[\sum_{i \in \mathcal{D}_l^b} \mathbb{1}_{y_{ic}=1} + \sum_{i \in \mathcal{D}_l^b} u_i \mathbb{1}_{q_{ic} > \tau} \right] \tag{11}$$

3.4. Overall training procedure

The overall training procedure of the proposed method is listed in Algorithm 1. Initially, supervised training step is performed on the ensemble (Fig. 3), where, each of the CNN classifier in the ensemble is trained independently with only the labeled data for a particular number of epochs (e.g., 50 epochs). This supervised pretraining stage is necessary to get reliable pseudo-labels for the unlabeled images. Without this supervised training, the calculated pseudo-labels will be noisy and wont help with semi-supervised training. After this initial supervised training stage, unlabeled samples are added with the training set, and then semi-supervised training is performed. In each training iteration, we randomly sample a mini-batch of size b which contains both labeled and unlabeled images. For each of the unlabeled image in this mini-batch, its pseudo-label and its weight are calculated based on the ensemble classifier using Eqs. (7) and (9) respectively (Fig. 4). In addition, the class weights also calculated for each mini-batch using Eq. (10). These pseudo-labels, instance weights and class weights are then used to update each of the CNN in the ensemble by the use of the loss function defined in Eq. (8) (Fig. 5). In the testing stage, the trained ensemble classifier is used to predict the label of each test image using Eq. (7).

3.5. CNN architectures

CNNs are widely used for image classification because of their superior performance over traditional machine learning approaches. CNN architectures usually contain a set of layers such as convolutional layers, non-linearity layers, pooling layers and fully connected layers. Although the proposed semi-supervised approach is not limited to a particular CNN architecture, we consider ResNet (He, Zhang, Ren, & Sun, 2016) and DenseNet (Huang, Liu, van der Maaten, & Weinberger, 2017) based architectures as they are widely used.

One of the main problem associated with training deep CNN architectures is the problem with vanishing/exploding gradients. When the CNN are trained with backpropagation, due to the multiplication of partial derivatives the gradients at the early layers of the network may close to zero or become very large value, which makes the learning harder. In order to overcome this, Residual Network (ResNet) introduces skip-connections–shortcut-connections which feed the output of a particular layer to later layers in the network that are not directly adjacent to the layer from which the output is originated. In this way, the gradients are easily propagated to the early layers in the backpropagation stage, and hence, it makes the training easy. In this work, we use two ResNet based CNN architectures, namely, ResNet-10 and ResNet-18, which contains 10 and 18 layers respectively. These two architectures are illustrated in Table 2.

Unlike ResNet, DenseNet connects each layer to every other layer, and therefore, further increases the gradient flow from the last layers to the early layers of the network. DenseNet reportedly show better performance than ResNet in Huang et al. (2017). In this work, we use a standard DenseNet-121 architecture in addition to the ResNet-based architectures.

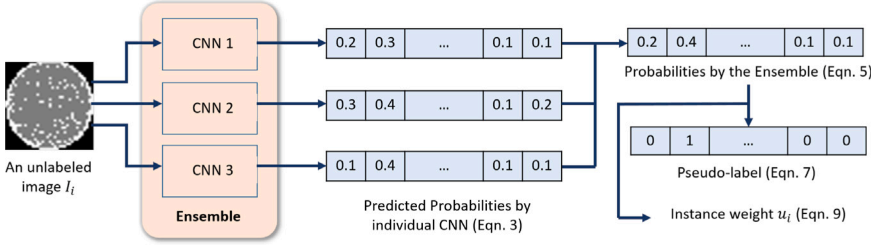

Fig. 4. Pseudo-label and instance weight calculations for an unlabeled image.

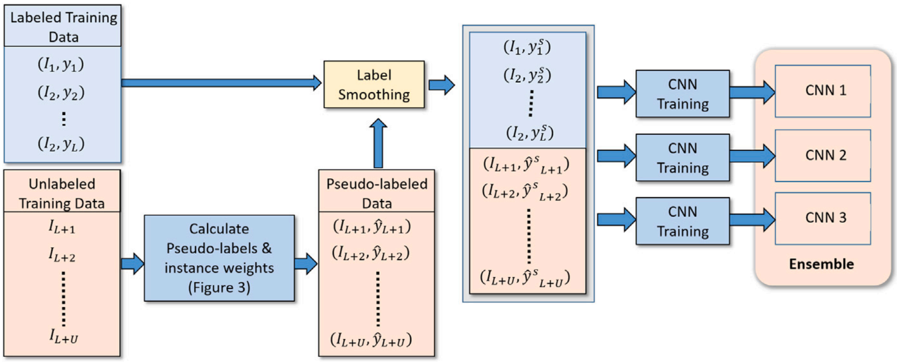

Fig. 5. Semi-supervised training at a particular iteration (L and U indicate the number of labeled and unlabeled data respectively).

Table 2
The detail of the ResNet architectures used in this work. 'Conv' indicates the convolutional layers. Inside the brackets are the shape of a residual block, and outside the brackets is the number of stacked blocks on a stage.

Layer name	Output size	ResNet-10	ResNet-18
conv1	$48 \times 48 \times 64$	7 × 7, 64, stride 2	
conv2	$24 \times 24 \times 64$	3 × 3, max pool, stride 2	
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3	$12 \times 12 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4	$6 \times 6 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5	$3 \times 3 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
Average pool	$1 \times 1 \times 512$	1 × 1 average pool	
Fully connected	9	512 × 9 fully connections	
softmax	9		

4. Experiments and results

In this section we first explain the dataset, experimental settings and the evaluation criteria, and then report and discuss the results.

4.1. Dataset

The proposed method is evaluated on the publicly available *WM-811K* dataset,[2] which contains a total number of 811,457 samples obtained from a real-world fabrication process. Among these samples only a subset of 172,950 samples were annotated by human experts; Each annotated sample in this subset is associated with an image-level label belonging to one of the nine predefined classes. Fig. 1 shows examples from each class. The class label distribution of this labeled set is highly imbalanced as summarized in Table 3.

4.2. Network training details

Stochastic Gradient Descent with an initial learning rate of 0.003 was used to train the CNN models. The settings used for training these architectures such as milestones, number of epochs, etc. are listed in Table 4. For example, for the fully-supervised baseline with ResNet architecture, a batch size of 256 is used, the initial learning rate was set to 0.003, and it was reduced by a factor of 10 at the end of the 50th and 100th epochs respectively, while the total number of epochs was set to 125. These parameters were selected based on some preliminary experiments. As explained in Section 3.4 for the semi-supervised approaches, for example, with the ResNet architecture, first the network was trained with only the labeled data for the first 50 epochs, and then for the next 100 epochs semi-supervised training was performed as given in Table 4. The weights of the CNN architectures are initialized by the use of ImageNet pretrained models.

4.3. Experimental setting

We follow the experimental setting of Kahng and Kim (2021) to have a fair comparison with Kahng and Kim (2021). The WBM images were resized to 96 × 96 pixels, and were normalized to have zero mean and unit standard deviation before they were given to the network. Random horizontal and vertical flipping were used as data augmentation at the training stage, and no data augmentation was used at the testing time. For each CNN architecture, a dropout rate of 50% was used to reduce overfitting. F1 score was used as the evaluation measure as it is widely used for this dataset. Each experiment was repeated three times and the mean and the standard deviations of the F1 scores obtained over these iterations are reported. In each iteration, 20% of the labeled data was used as the test set, and p% of the remaining labeled data was used for training. In each iteration, these data were randomly sampled from the labeled training set. Following Kahng and Kim (2021), p was changed from 5% to 100%. In the case of semi-supervised training, in addition to the p% of the labeled data, we randomly sampled 200,000 images from the unlabeled set and used that as the unlabeled data for training.

[2] https://www.kaggle.com/qingyi/wm811k-wafer-map

Table 3
WM-811K dataset contains a total of 811,457 (172,950 labeled and 638,507 unlabeled) wafer bin map images.

Class name	Labeled									Unlabeled
	Center	Donut	Edge-Loc	Edge-Ring	Loc	Random	Scratch	Near-Full	None	
No. of images	4294	555	5189	9680	3593	886	1193	149	147,431	638,507

Table 4
Training parameters for different network architectures. (FS — fully supervised, SS — semi-supervised).

CNN Architecture	Type	Batch size	Learning rate	No. of epochs	Milestones
ResNet	FS	256	0.003	125	[50, 100]
	SS	256	0.003	150	[125]
DenseNet	FS	128	0.003	90	[50, 75]
	SS	128	0.003	150	[125]

Table 5
Performance comparison of fully supervised (FS) vs. semi-supervised (SS) approaches with and without ensemble. K represents the size of the ensemble.

K	FS/SS	F1 score for different percentage (p%) of training samples per class				
		5% 6,918 imgs	10% 13,836 imgs	25% 34,590 imgs	50% 69,180 imgs	100% 138,360 imgs
ResNet-10						
1	FS	.760 ± .006	.799 ± .004	.839 ± .003	.859 ± .002	.870 ± .001
	SS	.810 ± .003	.847 ± .003	.881 ± .001	.885 ± .003	.894 ± .005
3	FS	.768 ± .001	.810 ± .002	0.847 ± .002	0.869 ± .001	.874 ± .002
	SS	**.823 ± .001**	**.857 ± .002**	**.887 ± .001**	**.897 ± .003**	**.900 ± .001**
ResNet-18						
1	FS	.800 ± .006	.827 ± .007	.872 ± .001	.886 ± .004	.890 ± .004
	SS	.834 ± .002	.859 ± .002	.883 ± .003	.898 ± .005	.902 ± .001
3	FS	.819 ± .001	.841 ± .007	.885 ± .002	.893 ± .002	.902 ± .002
	SS	**.850 ± .003**	**.871 ± .002**	**.900 ± .001**	**.910 ± .001**	**.912 ± .001**
DenseNet-121						
1	FS	.730 ± .006	.780 ± .011	.842 ± .006	.864 ± .002	.872 ± .004
	SS	.836 ± .003	.852 ± .001	.883 ± .002	.898 ± .002	.906 ± .002
3	FS	.750 ± .002	.802 ± .001	.856 ± .002	.870 ± .003	.880 ± .003
	SS	**.843 ± .003**	**.868 ± .002**	**.895 ± .001**	**.910 ± .002**	**.914 ± .001**

4.4. Performance of the proposed semi-supervised approach

Table 5 compares the performance of the proposed semi-supervised approach compared to its fully supervised counterpart for different percentage of training data with and without ensembling and with different CNN architectures. Here, the label smoothing parameter is set to $\alpha = 0.1$.

There are mainly three observations from Table 5: (1) Increasing the amount of labeled training data improves the classification performance of both fully supervised and semi-supervised approaches regardless of the network architecture and the ensembling technique used. (2) When the percentage of labeled training data is fixed, the proposed semi-supervised approach gives significantly improved performance compared to its fully supervised version regardless of the network architecture and the ensembling technique used. (3) In both the cases of full-supervision and semi-supervision, when the labeled training data is fixed ensembling gives improved performance compared to without using it regardless of the network architecture used.

When the amount of labeled training data is increased it will help to improve the decision boundaries of different classes of the network and hence the classification results. In the case of semi-supervised training, increasing the amount of labeled data for training also helps to correctly identify the pseudo-labels of the unlabeled data, and hence it also improves the classification performance of the semi-supervised approaches. For example, increasing the amount of training data from 5% to 10% gives an improvement of the F1 score by ~ 0.04 with the fully-supervised approach, and gives an improvement of the F1 score by ~ 0.03 with the semi-supervised approach with ResNet-10 architecture regardless of ensembling technique used.

Significant improvements were obtained by the proposed approach compared to its fully-supervised version regardless of ensembling and the CNN architecture used when fixing the amount of labeled data for training. For example, when ResNet-10 was used as the backbone CNN architecture, the F1 score was improved from 0.870 to 0.894 when ensembling is not used, and the F1 score was improved from 0.874 to 0.900 when ensembling is used. When DenseNet is considered with only 5% of the labeled training data, the F1 score is improved from 0.750 to 0.843 by the proposed approach compared to its fully supervised version.

The proposed approach not only improves the classification performance over its fully supervised counterpart, but also it gives better performance with less amount of labeled training data. For example, when ResNet-18 was used, the fully supervised approach gives a F1 score of 0.902 when all the labeled training data is used. On the other hand, a similar F1 score of 0.900 was obtained with only 25% of the labeled training data with the proposed semi-supervised approach, proving its efficiency.

The results summarized in Table 5 also show that regardless of the CNN architecture used, ensembling gives improved performance compared to without using it. For example, F1 score is improved from 0.890 to 0.902 when fully supervised approach is considered with ResNet-18 architecture and with all the labeled training data.

Table 6
Effect of label smoothing and ensembling, when selecting the pseudo-labeled samples. α represents the label smoothing factor. No label smoothing is used when $\alpha = 0$. N and $F1$ indicate the number of samples selected, and their F1 scores for different values of thresholds τ.

τ	no ensemble ($K = 1$)						with ensemble ($K = 3$)					
	$\alpha = 0$		$\alpha = 0.1$		$\alpha = 0.2$		$\alpha = 0$		$\alpha = 0.1$		$\alpha = 0.2$	
	N	$F1$	N	$F1$	N	$F1$	N	$F1$	N	$F1$	N	$F1$
.98	32,335	.95	1	1	0	–	30,845	.97	0	–	0	–
.95	32,917	.94	72	1	0	–	31,660	.97	4	1	0	–
.90	33,342	.92	25,719	.97	0	–	32,288	.96	**25,882**	.99	0	–
.85	33,607	.91	32,421	.96	1068	.98	32,663	.95	31,577	.97	40	1
.80	33,816	.91	33,037	.94	26,308	.94	32,949	.94	32,315	.96	22,068	.99
.00	34,590	.88	34,590	.88	34,590	.88	34,590	.88	34,590	.88	34,590	.88

Table 7
Size of the ensemble vs. classification performance by the proposed semi-supervised approach when 10% of labeled data is used for training.

K	1	2	3	5
F1 score	$.859 \pm 0.002$	$.868 \pm 0.002$	$\mathbf{.871 \pm 0.002}$	$\mathbf{.871 \pm 0.002}$

Table 8
Effect of sample weighting (soft vs. hard) when selecting samples.
(a) Soft-weighting: F1 scores for different values of β and τ.

τ	β		
	50	30	10
.95	.860	.866	.814
.90	.867	**.871**	.774
.80	.697	.683	.514

(b) Hard-weighting: F1 scores for different threshold (τ) values

τ	F1 score
.95	.853
.90	.860
.85	.858

In addition, the F1 score is improved from 0.902 to 0.912 when the proposed approach is considered with the same architecture and with all the labeled training data.

4.5. Effect of ensembling and label smoothing

This section aims to show that both ensembling and label smoothing help to alleviate the problems with high-confident predictions. In Section 4.4 we already showed that ensembling leads to improved performance for both fully supervised and semi-supervised approaches. In this section, we conducted further experiments to investigate the effect of ensembling and label smoothing, and the results are reported in Table 6. In these experiments, a model is trained using only the labeled dataset for 50 epochs, and then it is tested on the test set. Different threshold values were applied on the obtained probabilities of the test images, and the F1 scores of the selected test images are reported in Table 6.

When no label smoothing (i.e., $\alpha = 0$) and no ensembling is used a large number of images were predicted with high probabilities; for example, 32,335 out of 34,590 images predicted with probabilities greater than 0.98 and their F1 score is 0.95, meaning that there are many wrongly classified images which are predicted with high confident. When semi-supervised learning is applied on these predicted pseudo-labels, the model can easily trained with noisy labeled data and leads to degrade in performance. When label smoothing is not used, over 97% (33,816 out of 34,590) images were selected with very high confident predictions ($\tau \geq 0.8$), and most of them belongs to wrong classification (F1 score of 0.91). On the other hand, when label smoothing is used without ensembling, the percentage of correct predictions is improved. For example, when $\alpha = 0.1$ and $\tau \geq 0.90$, 25,719 out of 34,590 images were selected with an F1 score of 0.97, which is better than the F1 score obtained without label smoothing. However, a stronger smoothing parameter (i.e., $\alpha = 0.2$) gives better F1 scores (of 0.98) but at the same time significantly reduces the number of images selected based on the thresholds.

Ensembling alone helps to reduce the problem with over confident predictions. For example, F1 score is improved from 0.95 to 0.97 when ensembling is used with $\tau = 0.98$. However, applying both label smoothing and ensembling helps more to overcome the problem of over confident predictions. For example, when both label smoothing ($\alpha = 0.1$) and ensembling are used and $\tau = 0.90$, 25,882 images were selected with a very high F1 score of 0.99.

Table 7 reports the F1 scores against different sizes of ensemble. F1 score improves when increasing the size of the ensemble from one to three, and the results saturates after that. Therefore, we fixed the ensemble size to three, unless otherwise specified. An improvement of 0.012 (from 0.859 to 0.871) is obtained in this experiment when increasing the size of the ensemble from one to three.

4.6. Effect of sample weighting

This experiment investigates how the soft sample weighting defined in Eq. (9) affects the classification performance, and proves experimentally that this soft-weighting scheme gives improved performance over a hard-weighting scheme.

The soft sample weighting defined by Eq. (9) contains two parameters, τ and β. Table 8 reports the F1 scores of different values of these parameters. The best F1 score is obtained when $\tau = 0.9$ and $\beta = 30$ compared to other values. Smaller values of τ (e.g., $\tau = 0.8$) leads to a large number of non-zero weights, even for many wrongly classified samples (Table 6), and hence, gives low F1 scores. On the other hand, very large values of τ (e.g., $\tau = 0.95$) often select only few, but correctly identified samples (Table 6). Therefore, $\tau = 0.9$ is a good trade-off between the number of samples selected and their correctness.

When β is considered, a large value of β, (e.g., $\beta = 50$) leads to a step-like function, and therefore, the weights are either closer to zero or one. On the other hand, a very small value of β (e.g., $\beta = 10$) will give similar weights for all the samples. Therefore, $\beta = 30$ is a good trade-off parameter, and gives better results when $\tau = 0.90$.

Table 8 reports the F1 scores when a hard-thresholding scheme is used, where, instead of determining the weights based on Eq. (9) we apply a weight of one for the image I_i when its probability to the pseudo-label (q_{ic}) exceeds the threshold τ, and set to zero otherwise. Table 8 reports the F1 scores for different threshold values. When comparing the results from Tables 8 and 8 it is clear that the soft-weighting scheme defined in Eq. (9) gives the best performance compared to the hard-weighting scheme.

Table 9
Comparison with Kahng and Kim (2021): Classification performance for different percentage of training data, and with ResNet-18 architecture.

Method	5%		10%		25%		100%	
	F1	Acc.	F1	Acc.	F1	Acc.	F1	Acc.
Self-supervised representation learning (Kahng & Kim, 2021)	.815	–	.839	–	.864	–	.897	–
Ladder networks(Kong & Ni, 2018, 2020)	.814	95.4	.823	95.5	.838	96.2	.863	96.8
Ours	**.850**	**97.4**	**.871**	**97.7**	**.900**	**98.1**	**.912**	**98.2**

Table 10
Comparison with the state-of-the-art approaches (F1 score and Accuracy).

Method	F1	Acc.
Hand-crafted features + SVM (Wu et al., 2015)	–	94.6
Decision tree ensemble (Piao et al., 2018)	–	90.5
Light-weight CNN (Tsai & Lee, 2020)	0.800	97.0
Convolutional Neural Nets (Batool et al., 2020)	0.900	98.0
Self-supervised representation learning (Kahng & Kim, 2021)	0.897	–
Ours (25% of labeled training data)	**0.900**	**98.1**
Ours (100% of labeled training data)	**0.914**	**98.2**

4.7. Comparison with the state-of-the-art approaches

In this section we compare the results obtained by our method with other state-of-the-art approaches and show that our results are the new state-of-the-art.

Table 9 compares our method with the Ladder-Network based method proposed in Kong and Ni (2018, 2020) and the recently proposed semi-supervised representation learning approach in Kahng and Kim (2021) for different percentage of labeled training data with ResNet-18 architecture. Our method significantly performs better than all of these approaches. For example, when 5% of labeled training data is considered, our method gives an improvement of 0.04 compared to Kahng and Kim (2021), Kong and Ni (2018, 2020) (0.850 vs. 0.815). Our method achieves an F1 score of 0.900 with only 25% of labeled training data, on the other hand, Kahng and Kim (2021) achieves an F1 score of 0.897, and Kong and Ni (2018, 2020) achieves an F1 score of 0.863, but with all the labeled training data, which proves the effectiveness of our approach. Our approach is not only simple, but also requires low memory and computational requirements compared to the Ladder Network based approach proposed in Kong and Ni (2018, 2020).

Table 10 compares our approach with other state-of-the-art approaches. Our approach achieves the state-of-the-art F1 score of 0.900 with only 25% labeled data for training, and our approach beats other state-of-the-art approaches, and establishes a new state-of-the-art F1 score of 0.914 when all the labeled training data is considered.

5. Conclusion

In this work we proposed a novel semi-supervised deep learning approach for the classification of WBM defect patterns. In our approach, we first train an ensemble of CNN classifiers, and use them to estimate the pseudo-label of each unlabeled image, and its weight, and then use these pseudo-labeled images and their corresponding weights to update each CNN in the ensemble. We showed that label smoothing and ensembling are the two key factors of the success of our proposed approach. In addition, we showed that our proposed approach establishes the new state-of-the-art result for WBM classification. The proposed approach can be applicable with any CNN architecture. The future work will focus on applying the proposed method for other image classification problems.

CRediT authorship contribution statement

Siyamalan Manivannan: Conceptualization, Methodology, Software, Validation, Writing – review & editing.

Data availability

A public dataset was used for the experiments.

References

Adly, F., Alhussein, O., Yoo, P. D., Al-Hammadi, Y., Taha, K., Muhaidat, S., et al. (2015). Simplified subspaced regression network for identification of defect patterns in semiconductor wafer maps. *IEEE Transactions on Industrial Informatics*, *11*(6), 1267–1276.

Adly, F., Yoo, P. D., Muhaidat, S., Al-Hammadi, Y., Lee, U., & Ismail, M. (2015). Randomized general regression network for identification of defect patterns in semiconductor wafer maps. *IEEE Transactions on Semiconductor Manufacturing*, *28*(2), 145–152.

Arazo, E., Ortego, D., Albert, P., O'Connor, N. E., & McGuinness, K. (2020). Pseudo-labeling and confirmation bias in deep semi-supervised learning. arXiv:1908.02983.

Batool, U., Shapiai, M. I., Fauzi, H., & Fong, J. X. (2020). Convolutional neural network for imbalanced data classification of silicon wafer defects. In *IEEE International Colloquium on Signal Processing and Its Applications* (pp. 230–235).

Chang, C.-Y., Li, C., Chang, J.-W., & Jeng, M. (2009). An unsupervised neural network approach for automatic semiconductor wafer defect inspection. *Expert Systems with Applications*, *36*(1), 950–958.

Chien, C.-F., Hsu, S.-C., & Chen, Y.-J. (2013). A system for online detection and classification of wafer bin map defect patterns for manufacturing intelligence. *International Journal of Productions Research*, *51*(8), 2324–2338.

Chien, J.-C., Wu, M.-T., & Lee, J.-D. (2020). Inspection and classification of semiconductor wafer surface defects using CNN deep learning networks. *Applied Sciences*, *10*(15).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Hsu, C.-Y. (2015). Clustering ensemble for identifying defective wafer bin map in semiconductor. *Mathematical Problems in Engineering*.

Hsu, C.-Y., & Chien, J.-C. (2020). Ensemble convolutional neural networks with weighted majority for wafer bin map pattern classification. *Journal of Intelligent Manufacturing*, 1–14.

Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2261–2269).

Jeong, Y.-S., Kim, S.-J., & Jeong, M. K. (2008). Automatic identification of defect patterns in semiconductor wafer maps using spatial correlogram and dynamic time warping. *IEEE Transactions on Semiconductor Manufacturing*, *21*(4), 625–637.

Jin, C., 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*.

Kahng, H., & Kim, S. B. (2021). Self-supervised representation learning for wafer bin map defect pattern classification. *IEEE Transactions on Semiconductor Manufacturing*, *34*(1), 74–86.

Kong, Y., & Ni, D. (2018). Semi-supervised classification of wafer map based on ladder network. In *IEEE International Conference on Solid-State and Integrated Circuit Technology* (pp. 1–4).

Kong, Y., & Ni, D. (2020). A semi-supervised and incremental modeling framework for wafer map classification. *IEEE Transactions on Semiconductor Manufacturing*, *33*(1), 62–71.

Li, T.-S., & Huang, C.-L. (2009). Defect spatial pattern recognition using a hybrid SOM–SVM approach in semiconductor manufacturing. *Expert Systems with Applications*, *36*(1), 374–385.

Liao, C.-S., Hsieh, T.-J., Huang, Y.-S., & Chien, C.-F. (2014). Similarity searching for defective wafer bin maps in semiconductor manufacturing. *IEEE Transactions on Automation Science and Engineering*, *11*(3), 953–960.

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), 1479–1486.

Liukkonen, M., & Hiltunen, Y. (2018). Recognition of systematic spatial patterns in silicon wafers based on SOM and K-means. *Int. Conf. Math. Modell.*, *51*(2), 439–444.

Nakazawa, T., & Kulkarni, D. V. (2018). Wafer map defect pattern classification and image retrieval using convolutional neural network. *IEEE Transactions on Semiconductor Manufacturing*, *31*(2), 309–314.

Nguyen, A. M., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *IEEE Conf. Comput. Vis. Pattern Recogn.*, 427–436.

Palma, F. D., Nicolao, G. D., Miraglia, G., Pasquinetti, E., & Piccinini, F. (2005). Unsupervised spatial pattern classification of electrical-wafer-sorting maps in semiconductor manufacturing. *Pattern Recognition Letters*, *26*(12), 1857–1865.

Piao, M., Jin, C. H., Lee, J. Y., & Byun, J.-Y. (2018). Decision tree ensemble-based wafer map failure pattern recognition based on radon transform-based features. *IEEE Transactions on Semiconductor Manufacturing*, *31*(2), 250–257.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2818–2826).

Tsai, T.-H., & Lee, Y.-C. (2020). A light-weight neural network for wafer map classification based on data augmentation. *IEEE Transactions on Semiconductor Manufacturing*, *33*(4), 663–672.

Tulala, P., Mahyar, H., Ghalebi, E., & Grosu, R. (2018). Unsupervised wafermap patterns clustering via variational autoencoders. In *2018 International Joint Conference on Neural Networks* (pp. 1–8).

Van Engelen, J., & Hoos, H. (2020). A survey on semi-supervised learning. *Machine Learning*, *109*, 373–440.

Wang, R., & Chen, N. (2020). Defect pattern recognition on wafers using convolutional neural networks. *Quality and Reliability Engineering International*, *36*(4), 1245–1257.

Wang, C.-H., Kuo, W., & Bensmail, H. (2006). Detection and classification of defect patterns on semiconductor wafers. *IIE Transactions*, *38*(12), 1059–1068.

Wu, M.-J., Jang, J.-S. R., & Chen, J.-L. (2015). Wafer map failure pattern recognition and similarity ranking for large-scale data sets. *IEEE Transactions on Semiconductor Manufacturing*, *28*(1), 1–12.

Xie, Q., Luong, M.-T., Hovy, E., & Le, Q. V. (2020). Self-training with noisy student improves ImageNet classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, X., Song, Z., King, I., & Xu, Z. (2021). A survey on deep semi-supervised learning. CoRR abs/2103.00550 arXiv:2103.00550.