Semi-supervised Wafer Map Pattern Recognition using Domain-Specific Data Augmentation and Contrastive Learning

Hanbin Hu
Electrical and Computer Engineering
University of California
Santa Barbara, CA 93105
hanbinhu@ucsb.edu

Chen He
NXP Semiconductors
Austin, TX 78735
chen.he@nxp.com

Peng Li
Electrical and Computer Engineering
University of California
Santa Barbara, CA 93105
lip@ucsb.edu

***Abstract*—Wafer map pattern recognition is instrumental for detecting systemic manufacturing process issues. However, high cost in labeling wafer patterns renders it impossible to leverage large amounts of valuable unlabeled data in conventional machine learning based wafer map pattern prediction. We proposed a contrastive learning framework for semi-supervised learning and prediction of wafer map patterns. Our framework incorporates an encoder to learn good representation for wafer maps in an unsupervised manner, and a supervised head to recognize wafer map patterns. In particular, contrastive learning is applied for the unsupervised encoder representation learning supported by augmented data generated by different transformations (views) of wafer maps. We identified a set of transformations to effectively generate similar variants of each original pattern. We further proposed a novel *rotation-twist* transformation to augment wafer map data by rotating each given wafer map for which the angle of rotation is a smooth function of the radius. Experimental results demonstrate that the proposed semi-supervised learning framework greatly improves recognition accuracy compared to traditional supervised methods, and the rotation-twist transformation further enhances the recognition accuracy in both semi-supervised and supervised tasks.**

***Keywords*—wafer map pattern recognition, semi-supervised learning, contrastive learning.**

I. Introduction

As the crucial step of turning a chip design into a real product, robust semiconductor manufacturing process shall manifest high performance and yield of integrated circuits (ICs). However, the manufacturing process may suffer from different sources of systematic issues, such as stress cracking in chemical-mechanical polishing or contact to gate misalignment caused by a broken photo tool. Therefore, it is required that the manufacturing issues can be efficiently recognized and fixed; otherwise, the yield can be severely reduced and the production process quality is compromised.

One typical approach to detect such systematic manufacturing issues is via wafer map pattern recognition, which observes the die failure pattern on the wafer. Fig. 1 gives a few commonly-seen wafer map patterns, where each green pixel indicates a good die which passes all wafer tests, and each yellow pixel indicates a bad die which fails any wafer test. Different wafer map patterns might indicate different systematic manufacturing issues for a specific technology node.

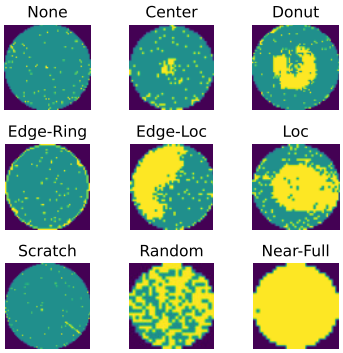


Fig. 1: Wafer map patterns.

Certain wafer patterns' implications might be straightforward and technology independent. For instance, the scratch failure pattern may imply some unexpected sharp edges touching and slipping on the surface of the wafer during the manufacturing or wafer test process. Regardless, being able to identify the wafer pattern quickly and automatically is important for manufacturing process issue detection and root cause analysis.

In order to recognize such failures on the fly, several machine learning techniques are investigated to automate the wafer map pattern recognition. [1], [2] applied support vector machine (SVM) to recognize wafer map patterns with pre-processed features using Radon transformation and geometric feature extraction. In recent years, [3]–[6] leveraged the strong image recognition power from convolution neural networks to identify wafer map patterns.

However, there are two major challenges which are not yet fully addressed by the previous works. First, it requires huge manual effort to correctly label a large wafer map dataset. If a wafer map is unlabeled, it cannot be utilized in the traditional supervised learning setting, severely reducing the effective number of training samples for learning. On the other hand, the unlabeled data may contain unrecognized patterns, which may provide extra information for wafer pattern learning. [7] proposed an unsupervised approach to recognize wafer map patterns using a set of recognizers trained by generative adversarial networks (GAN); however, the learning process of the multiple recognizers was heavily guided and tuned

manually. The second main challenge comes from that fact that wafer map pattern data is typically highly imbalanced. Stable manufacturing processes mostly produce wafer maps that show no patterns at all. The ones with patterns only constitute a small chunk in the entire dataset. Such imbalance may bias a trained recognizer towards making only correct decisions on the dominant wafers without a pattern in the dataset while more important problematic patterns are mis-predicted, causing systematic failure escape. [3], [6] suggested using an auto-encoder (AE) architecture to augment under-represented patterns; however, such augmentation still requires the data label, which may be unrealistic in many cases.

In order to fully use the unlabeled data for wafer map pattern recognition, we proposed a semi-supervised learning framework to first learn a good representation of all wafer map patterns presented in a (potentially large) unlabeled dataset, and then recognize wafer map patterns in a supervised manner with a small labeled dataset. Our methodology has been motivated by the recent development in contrastive learning for unsupervised representation learning in the machine learning community. Contrastive learning is a framework to learn good representation of given data in several applications like image recognition and natural language processing [8]–[10]. SimCLR [9] is one contrastive learning technique which achieves comparable or even better performance compared to its supervised counterpart for image recognition tasks demonstrated using the popular ImageNet dataset [11]. The representation is learnt in a self-supervised manner via comparison, i.e., different transformations (views) of the original data are compared to extracted an informative representation. The contrastive learning has been explored in the wafer map pattern detection field in [12], while no novel domain-specific transformations were proposed.

This paper presents a domain-specific application of contrastive learning for wafer pattern recognition. While the existing contrastive learning techniques have primarily focused on conventional image recognition and natural language processing tasks, the unique characteristics of the wafer pattern recognition task presents new challenges and opportunities. First, we investigate the relevance of the transformations proposed in the literature, which act as a mechanism for data augmentation, and identify a near-optimal subset of transformations that are well-suited for meaningful characterization of similarities and dissimilarities of practical wafer pattern data. Furthermore, we study rotation-based transformations, which are rarely employed in conventional image data analysis. We propose a novel rotation operation that transforms a given wafer pattern into a *similar* pattern by performing non-uniform rotations of the dies on the wafer for which the angle of rotation is a smooth function of the radius. Our proposed new *rotation-twist* transformation acts as a domain-specific data augmentation technique and enables automated generation of high-volumes of similar wafer data while retaining the structure of the original wafer pattern. Experimental results demonstrate that the proposed semi-supervised learning framework greatly improves recognition accuracy compared to traditional supervised methods, and the *rotation-twist* transformation further enhances the recognition accuracy in both semi-supervised and supervised tasks.

II. Semi-Supervised Wafer Pattern Recognition

A. Problem Formulation

A wafer map of width W and height H can be denoted by $\mathbf{x} \in \mathcal{X} = \{-1, 0, 1\}^{W \times H}$, where 0 is used for good dies, 1 is used for bad dies, and -1 indicates that there is no die in the current location.

Consider a labeled dataset $\mathcal{D}_L = \left\{\cdots, \left(\mathbf{x}^{(l)}, y^{(l)}\right), \cdots\right\}$ with N_L labeled samples with each $\mathbf{x}^{(l)}$ standing for the l-th wafer map and each $y^{(l)} \in \mathcal{Y}$ for its corresponding pattern. In addition, since labeling all wafer maps requires huge manual effort, we consider another unlabeled dataset $\mathcal{D}_U = \left\{\cdots, \mathbf{x}^{(k)}, \cdots\right\}$ with N_U unlabeled samples. Typically, $N_U \gg N_L$.

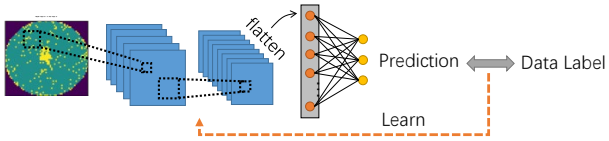
To learn a wafer map recognizer, we optimize a model represented by $f : \mathcal{X} \to \mathcal{Y}$, which is parameterized by $\boldsymbol{\theta}$, to minimize the difference between the recognizer prediction $\hat{y} = f(\mathbf{x}; \boldsymbol{\theta})$ and the true data label y governed by a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ as follows.

$$\theta^* = \arg\min \mathbb{E}\left[L\left(f(\mathbf{x}; \boldsymbol{\theta}), y\right) \mid \mathcal{D}_L, \mathcal{D}_U\right] \tag{1}$$

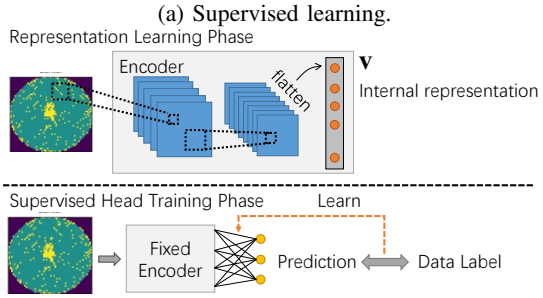
Note that in supervised learning, the model prediction $\hat{y}$ is directly compared with the true label y via the loss function. Therefore, the unlabeled dataset $\mathcal{D}_U$ cannot be used to compute the loss function, wasting a large number of unlabeled data in the wafer map pattern recognition. If the model is constructed using a neural network with $\boldsymbol{\theta}$ as its model weights, as shown in Fig. 2a, only the comparison between model prediction and data label is used to tune the model weights $\boldsymbol{\theta}$ for the entire neural network, resulting in a huge search space and a potentially over-fit recognizer.

B. Semi-Supervised Learning

In order to fully utilize the unlabeled dataset $\mathcal{D}_U$, a semi-supervised learning framework is introduced here. Instead of directly learning the recognizer model f, we can rewrite the recognizer as a composition of two functions $f = h_s \circ v$. Here $v : \mathcal{X} \to \mathcal{V}$, parameterized by $\boldsymbol{\theta}_e$, encodes the origin



(a) Supervised learning.



(b) Semi-supervised learning.

Fig. 2: Supervised and semi-supervised learning comparison.

wafer map into an internal representation space $\mathcal{V}$, and $h_s : \mathcal{V} \to \mathcal{Y}$, parameterized by $\boldsymbol{\theta}_s$, is a head predicting the data label using the internal representation. Correspondingly, the original model learning procedure is split into 2 phases as follows, which is also shown in Fig. 2b.

$$f(\mathbf{x};\boldsymbol{\theta}) = h_s\left(v\left(\mathbf{x};\boldsymbol{\theta}_e\right);\boldsymbol{\theta}_s\right) \tag{2}$$

A good representation $\mathbf{v}$ of the original wafer map is learnt using unlabeled data $\mathcal{D}_U$, of a potentially large size, in the first phase so that it can ease the downstream task of supervised head h_s training. How to learn a good representation from unlabeled data depends on what property the representation is expected to extract and possess. For example, a typical consideration in AE is that the representation should compress all information to reconstruct the original data. In contrastive learning, which is further described in Section III, we would like ensure that representations of similar wafer maps are close to each other. The resulting proxy task to learn good representation is typically different from (1), eliminating the needs to use the same loss function requiring labeled data and enabling unsupervised learning for wafer map pattern recognition using $\mathcal{D}_U$.

Note that in the second phase, the encoder parameters $\boldsymbol{\theta}_e$ is fixed, and the labeled data is only used to learn the supervised head model h_s to recognize wafer map patterns. With a fixed $\boldsymbol{\theta}_e$, the parameter space for $\boldsymbol{\theta}$ is greatly reduced, accelerating the optimization process for the supervised head parameters $\boldsymbol{\theta}_s$. In addition, fixing the internal representation avoids disrupting the learnt representation during the learning process of the second phase. If the encoder weights are tuned, it is highly likely that the learnt representation is overwritten with the new information provided by the labeled data, degrading the model into a simple supervised learning model.

III. Semi-supervised Representation Learning via Contrastive Learning Framework

A. Learning Similarity via Data Augmentation

Recently, a novel self-supervised learning framework called contrastive learning suggests extracting a good data representation by learning the similarity among samples [8]–[10]. Consider two wafer maps $\mathbf{x}_1$ and $\mathbf{x}_2$. If they are similar (sharing the same wafer map pattern), their internal representations $\mathbf{v}_1$ and $\mathbf{v}_2$ should be "close" to each other.

However, as most data labels are unavailable, the similarity among samples is not revealed explicitly by data labels. Instead, contrastive learning introduces transformations to create similar variants of each given sample and maintain similarities among these variants as part of the unsupervised representation learning. Incorporating these augmented data leads to a larger dataset with more samples. Consider a transformation $T : \mathcal{X} \to \mathcal{X}$ sampled from a transformation set $\mathcal{T}$. The transformations are well selected to make each transformed variant $\tilde{\mathbf{x}}$ similar (sharing the same wafer map pattern) with the original sample $\mathbf{x}$. Therefore, for two transformations T and T', the resulting internal representations

$$\mathbf{v} = v\left(T\left(\mathbf{x}\right);\boldsymbol{\theta}_e\right),\ \mathbf{v}' = v\left(T'\left(\mathbf{x}\right);\boldsymbol{\theta}_e\right) \tag{3}$$

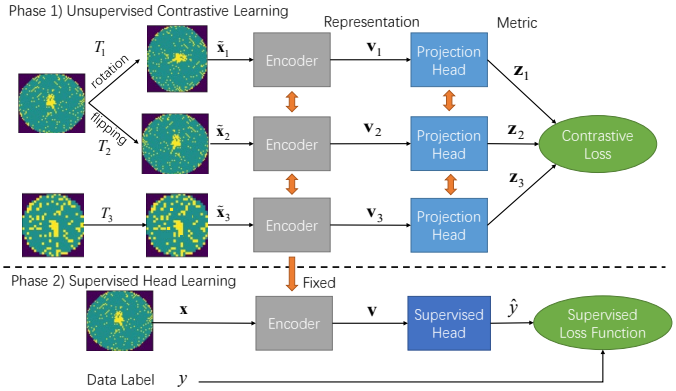


Fig. 3: Semi-supervised contrastive learning architecture.

should be made close to each other. The "closeness" between internal representations is governed by a *contrastive loss* in an unsupervised manner. For two similar samples, their corresponding internal representations should be close to each other, suggesting a small contrastive loss; otherwise, a large contrastive loss should be generated for dissimilar samples.

B. Semi-supervised Contrastive Learning Framework

1) Framework Overview: Fig. 3 gives a brief illustration of how semi-supervised contrastive learning works. As mentioned in Section II, the semi-supervised learning is conducted in two phases. For the first unsupervised contrastive learning phase, we carefully design a transformation set $\mathcal{T}$ to identify the similarity among samples. Each sample $\mathbf{x} \in \mathcal{X}$ is first transformed to multiple variants $\tilde{\mathbf{x}}$ in the original sample space $\mathcal{X}$. For simiplicity, let's say $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ are transformed through two randomly-selected transformations $T \sim \mathcal{T}$ (e.g., rotation) and $T' \sim \mathcal{T}$ (e.g., flipping); then they go through the same encoder to obtain their internal representations $\mathbf{v}$ and $\mathbf{v}'$ in the *internal representation space* $\mathcal{V}$; finally, the internal representation $\mathbf{v}$ is mapped into a *metric embedding space* $\mathcal{Z}$ through a projection head. The training of contrastive loss, requiring no data label, learns the similarity among samples to formulate a good internal representation via comparing the metric embeddings $\mathbf{z}$ from different variants of multiple samples in the metric embedding space $\mathcal{Z}$.

In the second supervised head training phase, the encoder is inherited and fixed from the previous phase to obtain the trained internal representation $\mathbf{v} \in \mathcal{V}$. An additional supervised head is added to make prediction $\hat{y}$ from $\mathbf{v}$. The loss function is defined by the true data labels over a small labeled dataset to guide the supervised head learning to predict the correct wafer map pattern.

During the inference stage, only the encoder and the supervised head are utilized for recognizing the patterns of the previously-unseen wafer maps, while The projection head is discarded after the first training phase.

2) Contrastive Learning Objective: Note that the internal representation v is employed in both training phases where two different learning objectives are targeted as shown in Fig. 3. We disentangle the unsupervised contrastive learning objective from that of the supervised head learning by introducing the projection head $h_p : \mathcal{V} \to \mathcal{Z}$ that is parameterized by $\boldsymbol{\theta}_p$ and

maps the internal representation v to the metric embedding space $\mathcal{Z}$ in which the contrastive loss is defined. This is more beneficial than defining the contrastive loss directly based on the internal representation v. This approach avoids overly constraining the internal representation learning towards the minimization of the contrastive learning loss and allows for more flexible learning of the internal representation to better support the supervised learning task in the second phase.

To this end, the "closeness" in the internal representation space we discussed in Section III-A shall be interpreted broadly. Two internal representations can be considered "close" as long as their mapped metric embeddings are close enough. Hence, the contrastive loss between two metric embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$ from the same sample within a batch of metric embeddings $\{\mathbf{z}_k\}_{k=1}^{B}$ can be defined as

$$L_{cl}\left(\mathbf{z}_i, \mathbf{z}_j\right) = -\log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_j/\tau\right)}{\sum_{k\in\{i\}'} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_k/\tau\right)}, \tag{4}$$

where τ is a temperature coefficient to characterize the relative distance among metric embeddings. Here $\{i\}'$ denotes the set complement $\{1, \cdots, B\} \setminus \{i\}$ excluding element i. To minimize the contrastive loss L_{cl}, the similarity (inner product) between metric embeddings $\mathbf{z}_i$ and $\mathbf{z}_j$ should be as large as possible, while the similarity (inner product) with other embedding $\mathbf{z}_k$ should become small, implying a good representation learnt by the encoder v.

Note that the metric embedding $\mathbf{z}$ is usually normalized with $\mathbf{z}/\left\|\mathbf{z}\right\|$ to avoid scaling issues among different sample views.

C. Semi-supervised Contrastive Learning Algorithm

With the contrastive learning framework described previously, Algorithm 1 summarizes the detailed procedure of the entire semi-supervised wafer pattern recognition algorithm. Note that for each sample $\mathbf{x}_k$, two parts of contrastive loss $L_{cl}\left(\mathbf{z}_k, \mathbf{z}'_k\right)$ and $L_{cl}\left(\mathbf{z}'_k, \mathbf{z}_k\right)$ are computed, as the two metric embeddings $\mathbf{z}_k$ and $\mathbf{z}'_k$ from the two views $\tilde{\mathbf{x}}_k$ and $\tilde{\mathbf{x}}'_k$ of the sample sample $\mathbf{x}_k$ are symmetric to each other in terms of contrastive loss computation.

Algorithm 1: Semi-supervised contrastive learning.

Input : Unlabeled dataset $\mathcal{D}_U$; Labeled dataset $\mathcal{D}_L$.
Output: Model prediction parameters $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_s$.
Hyperparameters: Transformation set $\mathcal{T}$, epochs T_U, batch size B_U, and temperature coefficient τ for unsupervised contrastive learning; Epochs T_L and batch size B_L for supervised head learning;

`/* unsupervised contrastive learning */`
for $t \leftarrow 1$ **to** T_U **do**
  for *a sampled minibatch* $\{\mathbf{x}_k\}_{k=1}^{B_U}$ *from* $\mathcal{D}_U$ **do**
    for $k \leftarrow 1$ **to** B_U **do**
      Sample transformations $T \sim \mathcal{T}$, $T' \sim \mathcal{T}$;
      $\tilde{\mathbf{x}}_k \leftarrow T\left(\mathbf{x}_k\right)$; $\tilde{\mathbf{x}}'_k \leftarrow T'\left(\mathbf{x}_k\right)$;
      $\mathbf{v}_k \leftarrow v\left(\tilde{\mathbf{x}}_k; \boldsymbol{\theta}_e\right)$; $\mathbf{v}'_k \leftarrow v\left(\tilde{\mathbf{x}}'_k; \boldsymbol{\theta}_e\right)$;
      $\mathbf{z}_k \leftarrow h_p\left(\mathbf{v}_k; \boldsymbol{\theta}_p\right)$; $\mathbf{z}'_k \leftarrow h_p\left(\mathbf{v}'_k; \boldsymbol{\theta}_p\right)$;
      Compute contrastive loss using $l_k \leftarrow -\log \frac{\exp\left(\mathbf{z}_k \cdot \mathbf{z}'_k/\tau\right)}{\sum_{i\in\{k\}'} \exp(\mathbf{z}_k \cdot \mathbf{z}_i/\tau) + \sum_{i=1}^{B_U} \exp\left(\mathbf{z}_k \cdot \mathbf{z}'_i/\tau\right)}$;
      Compute other contrastive loss using $l'_k \leftarrow -\log \frac{\exp\left(z_k \cdot \mathbf{z}'_k/\tau\right)}{\sum_{i=1}^{B_U} \exp\left(\mathbf{z}'_k \cdot \mathbf{z}_i/\tau\right) + \sum_{i\in\{k\}'} \exp\left(\mathbf{z}'_k \cdot \mathbf{z}'_i/\tau\right)}$;
    end
    $\mathcal{L}_U \leftarrow \frac{1}{2B_U} \sum_{k=1}^{B_U} \left(l_k + l'_k\right)$;
    Update $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_p$ to minimize $\mathcal{L}_U$;
  end
end
`/* supervised head learning */`
Fix encoder $\boldsymbol{\theta}_e$ and drop projection head $\boldsymbol{\theta}_p$;
for $t \leftarrow 1$ **to** T_L **do**
  for *a sampled minibatch* $\{\mathbf{x}_k, y_k\}_{k=1}^{B_L}$ *from* $\mathcal{D}_L$ **do**
    for $k \leftarrow 1$ **to** B_L **do**
      $\hat{y}_k \leftarrow h_s\left(v\left(\mathbf{x}_k; \boldsymbol{\theta}_e\right); \boldsymbol{\theta}_s\right)$;
      $l_k \leftarrow L\left(\hat{y}_k, y_k\right)$;
    end
    $\mathcal{L}_L \leftarrow \frac{1}{B_L} \sum_{k=1}^{B_L} l_k$;
    Update $\boldsymbol{\theta}_s$ to minimize $\mathcal{L}_L$;
  end
end
return *encoder parameters* $\boldsymbol{\theta}_e$ *and supervised head parameters* $\boldsymbol{\theta}_s$.

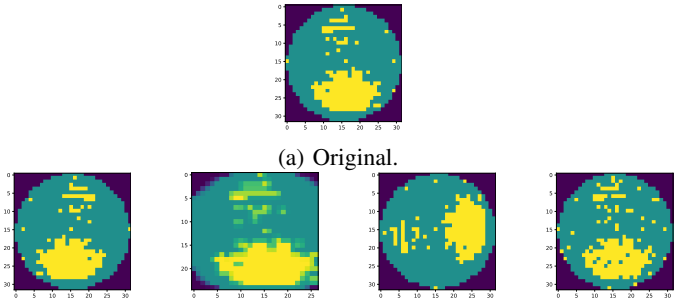
IV. Relevance of Existing Contrastive Learning Transformations for Wafer Pattern Recognition

A. Consideration for Domain Specific Transformations

One important component to determine the performance of the unsupervised contrastive learning is the transformation set $\mathcal{T}$. Transformations included in $\mathcal{T}$ suggest what kind of similarity should be maintained after the transformation, i.e., what similarity among samples should be learnt for a good representation mapped by the encoder. For example, in the image recognition task, if we consider only color distortion transformation without affine transformation in the contrastive learning, the dogs with different colors can be well coded in the internal representation space $\mathcal{V}$, however, the dogs with different sizes and locations in images may be considered distinct in $\mathcal{V}$. We consider three traits for a good transformation to be considered for the representation learning task as follows.

1) Suitability for the task: Obviously, the transformations under consideration should be suitable for the data under analysis. If the transformation cannot be applied to the data sample space $\mathcal{X}$, it should not be considered.

2) Capability to maintain similarity among samples: This is the most critical characteristic for a good transformation. The transformation should maintain the major sample property unchanged after transformation. Here, in our application, wafer map patterns should be kept the same, while the detailed good/bad die distribution in a wafer map can be varied by the transformations.



(a) Original.

(b) Flipping. (c) Cropping. (d) Rotation. (e) Noise.

Fig. 4: Several adopted transformations for wafer map pattern recognition. Note that only random horizontal flipping and random resized cropping were chosen in the SimCLR work [9] for traditional image recognition.

3) Flexibility to be randomized: Finally, the transformations should be easy to be randomized, adding more flexibility when transforming the details of samples, so that the similarity among samples can be fully captured by the transformations.

B. Existing Contrastive Learning Transformations

One of the most well-known works in contrastive learning, SimCLR [9], studied a large set of transformations for general image recognition tasks like ImageNet [11]. In particular, four transformations are recommended for standard image recognition tasks: 1) random horizontal flipping, 2) random resized cropping, 3) color jitter, and 4) Gaussian blur.

For the wafer map pattern recognition application, however, not all of the four transformations in SimCLR [9] are suitable. Since a single location x_{ij} in a wafer map may only contain three possible values $\{-1, 0, 1\}$, clearly, the Gaussian blur and the color jitter cannot be applied to the wafer map dataset. On the other hand, we argue that the other two transformations are suitable for the wafer map recognition task. 1) The *random horizontal flipping* augments the unlabeled wafer data by introducing symmetric variants of wafer maps without changing their patterns. 2) The *random resized cropping* crops a random portion of the wafer map and resize the cropped portion back to the original wafer size. It creates new variants of wafer data by extracting a local view of an original wafer map and re-projecting that onto the full wafer.

V. Proposed Domain-Specific Data Augmentation for Wafer Pattern Recognition

As mentioned in Section IV, we adopt *random horizontal flipping* and *random resized cropping* used in traditional image recognition [9] for wafer map pattern recognition, as shown in Fig. 4b and 4c, respectively. In addition, we propose three domain-specific transformations.

A. Random Wafer Rotation

Rotations are not suitable for traditional image analysis tasks in which images often maintain a proper orientation, e.g., an upside-down pedestrian is unlikely in a realistic visual scene. However, as illustrated in Fig. 4d, rotating a wafer does not alter its circular shape, but can change the orientation of the good and bad dies while maintaining the same basic wafer map pattern. To generate a variety of rotated wafer maps, we consider two random rotation schemes as follows. The first scheme rotates wafers with a degree randomly selected from a continuous range $[0°, 360°)$, denoted as "Continuous" in Table II. The second scheme rotates wafers with a degree randomly selected from a finite set $\{0°, 90°, 180°, 270°\}$, denoted as "Discrete" in Table II.

B. Random Die Noise

Adding Gaussian noise to images as a way of transformation was shown to be ineffective for traditional image recognition tasks [9]. Differently, as shown in Fig. 4e, we randomly flip good and bad dies at position (i, j) where a valid die ($x_{ij} \neq -1$) locates as follows.

$$\tilde{x}_{ij} = \begin{cases} 1 - x_{ij} & \text{with a probability of } p_n \\ x_{ij} & \text{with a probability of } 1 - p_n \end{cases} \quad (5)$$

With a small p_n value, we largely maintain the present wafer map pattern while introducing a certain degree of perturbation to the wafer. In our experiments, p_n is set to be 0.05.

C. Rotation-Twist Transformation

Although the previous domain-specific transformations extract good similarity among samples for contrastive learning, these transformations don't change the detailed shape significantly in wafer maps. For example, a scratch pattern cannot be curved no matter which transformation is adopted. With the detailed shape twisted, the relative positioning and the correlation among close dies in wafer maps are modified. Note that the detailed wafer map shapes may vary a lot even for the same wafer map pattern.

Here we propose a novel transformation to twist the detailed shape inside wafer maps while maintaining the pattern for recognition unchanged. Specifically, the proposed transformation performs non-uniform rotations of the dies on the wafer for which the angle of rotation is a smooth function of the radius. As shown in Fig. 5, the proposed transformation can transform a straight scratch into a curved scratch. In order to do so, three steps are executed to accomplish the entire twisting. First we randomly generate a smooth angle function to provide certain randomness to the transformation. Then, the wafer map is twisted via rotation based on the generated smooth angle function. Finally, the wafer map is regularized to make it valid. The details of each step are given below, and several examples are provided at the end of the section.

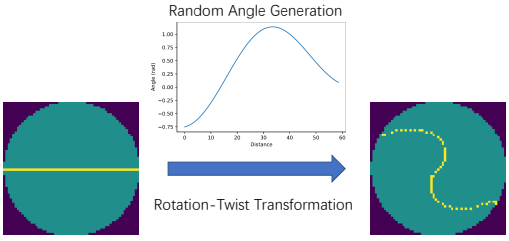
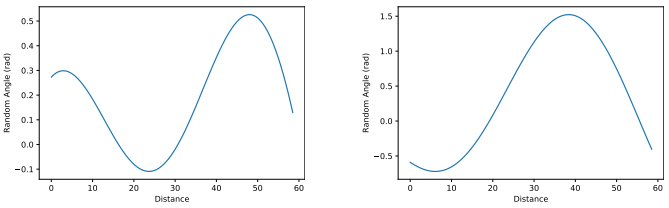


Fig. 5: Rotation-twist transformation illustration.



(a) Example 1. (b) Example 2.

Fig. 6: Random angle generation examples with $m = 3$.

1) Random Smooth Function Generation: To add a certain degree of randomness to the final wafer map twisting results, a random function is first generated, and then the detailed twist shape is based on the generated random smooth function. To avoid the wafer map pattern altered due to the abrupt function value change, we apply a Fourier-transform-based random function sampling method in [13] to generate a random smooth function as follows.

$$\theta(d) = a_0 + \sqrt{2}\sum_{j=1}^{m}\left[a_j \cos\frac{2\pi j d}{D} + b_j \sin\frac{2\pi j d}{D}\right], \quad (6)$$

where D is the sum of the wafer width and height, and m is the order of the Fourier transform. Each a_j and b_j is an independent Gaussian variable from $\mathcal{N}(0, 1/(2m+1))$. As shown in Fig. 6, the 2 examples of generated random function with $m = 3$ is relatively smooth within the radius of the wafer.

2) Twist via Rotation: After the random angle function is generated, the actual twisting is mainly executed via rotation. All the dies with the same radius to the center of the wafer map are rotated with an angle determined by the generated random angle function and its corresponding radius, ending up with the following equation to relocate the die at (i, j).

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} \cos\theta\left(\sqrt{i^2+j^2}\right) & -\sin\theta\left(\sqrt{i^2+j^2}\right) \\ \sin\theta\left(\sqrt{i^2+j^2}\right) & \cos\theta\left(\sqrt{i^2+j^2}\right) \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} \quad (7)$$

However, we may notice that the resulting location (i', j') is not guaranteed to be an integer. Here the nearest neighbor is taken by rounding the location to the closest integer grid. Therefore, we can map the goodness of a particular die at (i, j) in the original wafer map $\mathbf{x}$ to $(\lfloor i' \rceil, \lfloor j' \rceil)$ in the twisted wafer map $\tilde{\mathbf{x}}$ as follows.

$$\tilde{x}_{\lfloor i' \rceil \lfloor j' \rceil} = x_{ij} \quad (8)$$

3) Wafer Map Regularization: After twisting the wafer map, a few additional steps are proceeded to make sure the resulting wafer map is a valid one.

a) Maintain wafer map shape: First, we ensure that the shape of the twisted wafer map matches the original one by guaranteeing that the resulting wafer $\tilde{\mathbf{x}}$ doesn't have die at (i, j) where $x_{ij} = -1$.

b) Fill in Holes using Majority Votes: Secondly, according to (7) and (8), the resulting wafer map may have a few locations with no corresponding original wafer map locations. For those unfilled positions (holes) in the wafer map, we determine its die goodness via a majority vote of its surrounding 4 dies. The new wafer map is kept updating until no change occurs in two consecutive updates.

c) Final check: Finally, we set all the remaining unfilled-in locations with good dies. With that, a valid randomly-twisted wafer map $\tilde{x}$ is generated.

Algorithm 2 summarizes the entire procedure for the *rotation-twist* transformation.

Algorithm 2: Rotation-Twist transformation flow.

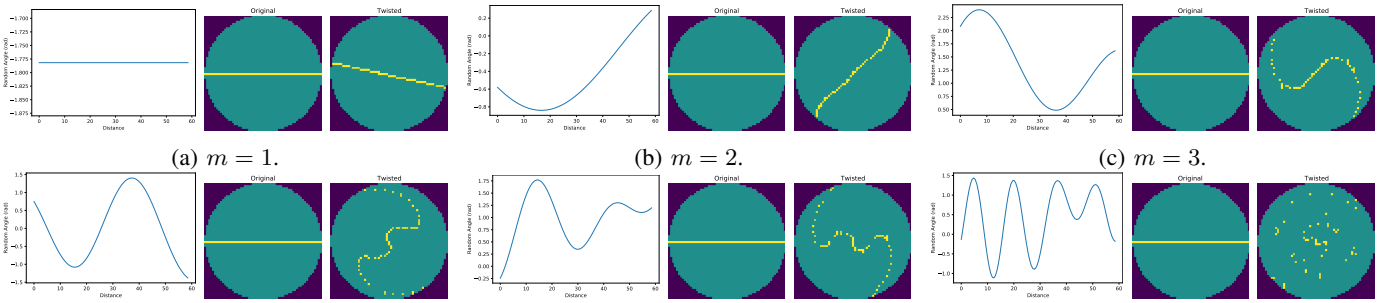
Input : A wafer map $\mathbf{x}$.
Output: The transformed wafer map $\tilde{\mathbf{x}}$.
Hyperparameters: Order m for random smooth angle function generation;

```
/* Random smooth function generation */
```
Sample $a_i \sim \mathcal{N}(0, 1/(2m+1))$ for $i = 0, \cdots, m$;
Sample $b_i \sim \mathcal{N}(0, 1/(2m+1))$ for $i = 1, \cdots, m$;
Construct the angle function $\theta(d)$ using (6);
```
/* Twist via rotation */
```
Generate a new wafer map $\tilde{\mathbf{x}}$ using (7) and (8);
```
/* Wafer map regularization */
```
Let $\tilde{x}_{ij} = -1$ where $x_{ij} = -1$;
do
  Update undefined die $\tilde{x}_{ij}$ with a majority vote using the surrounding four dies $\tilde{x}_{i+1,j}$, $\tilde{x}_{i-1,j}$, $\tilde{x}_{i,j+1}$, and $\tilde{x}_{i,j-1}$;
while *no update in* $\tilde{\mathbf{x}}$ *is observed*;
Let all remaining undefined dies as good dies;
return *The transformed wafer map* $\tilde{\mathbf{x}}$.

D. Rotation-Twist Transformation Examples

As the twisting effect of the *rotation-twist* transformation is determined by the generated random smooth angle function, we first check how different orders m affect the resulting twisted wafer maps. Fig. 7 gives a few examples of the resulting wafer maps after applying the *rotation-twist* transformation with different orders to the same wafer map with a single line pattern. For each order, the first plot is the generated random angle function, and the rest two images are the original wafer map and the twisted one. When $m = 1$, the proposed transformation is degraded into simple rotation, which provides no additional information to contrastive learning; when $m = 2$, the scratch in the wafer map starts to be twisted, but in a relatively small scale. As the order grows, more twisted detailed shapes can be observed in the wafer maps, which changes the relative positioning of close dies in wafer maps while maintaining the wafer map patterns when $m = 3$ or $m = 4$. However, if we keep increasing the order, we can see the wafer map pattern can be destroyed as higher frequency components are added into random angle function, resulting abrupt function changes like $m = 10$.

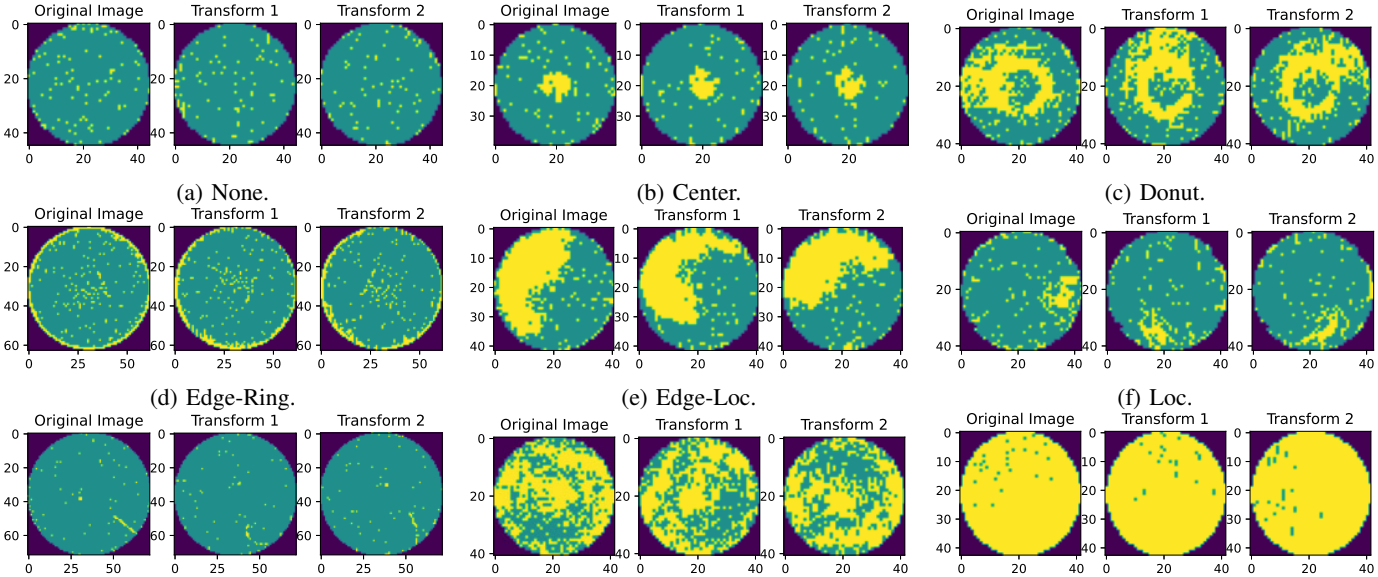
Furthermore, Fig. 8 gives several wafer map examples after applying the proposed rotation-twist transformation for all the patterns under consideration with an order of $m = 3$. For each pattern in Fig. 8, the first image is the original wafer map in the dataset, the second and the third images are two examples of the twisted views. In general, it can be observed that all the



(a) $m = 1$. (b) $m = 2$. (c) $m = 3$.

(d) $m = 4$. (e) $m = 5$. (f) $m = 10$.

Fig. 7: Rotation-Twist transformation with different order m.



(a) None. (b) Center. (c) Donut.

(d) Edge-Ring. (e) Edge-Loc. (f) Loc.

(g) Scratch. (h) Random. (i) Near-Full.

Fig. 8: Rotation-Twist transformation examples for all the considered wafer map patterns with $m = 3$.

wafer map patterns have been kept after the transformation while the detailed shape in each wafer map is significantly changed. For example, in Fig. 8g, although we can treat all the three wafer maps as "Scratch" pattern, the original scratch is perpendicular to the wafer edge, while the twisted scratches are not. In addition, the generated scratches are a bit curved compared to the original one. In addition, the localized die failure patterns are significantly changed after transformation, such as the "Edge-Loc" and "Loc" patterns in Fig. 8e and 8f. These wafer map examples demonstrate the effectiveness of the proposed transformation to twist wafer maps while maintaining their patterns.

VI. Experimental Results

A. Experimental Settings

1) Methods under Comparison: We compared our proposed semi-supervised contrastive learning method with two sets of baseline methods. The first set of baseline methods mainly adopts a supervised learning methodology, including the support vector machine (SVM) [1], and the convolutional neural network (CNN) architectures, which are widely adopted in [3]–[6]. The second baseline method adopts the semi-supervised contrastive learning method with *random horizontal flipping* and *random resized cropping* chosen as transformations. These two transformations are the only transformations used in SimCLR [9] that are suitable for wafer pattern recognition. Hence, the second baseline corresponds to the existing state-of-the-art contrastive learning technique when applied to wafer map data.

We consider two variants of our proposed semi-supervised contrastive learning method. The first variant adopts four adapted transformations: *random horizontal flipping*, *random resized cropping*, *random wafer rotation*, and *random die noise*, denoted by "4TCLWMPR" (4 transformations in contrastive learning for wafer map pattern recognition). The second variant includes the *rotation-twist* transformation in addition to the four transformations included in the first variant, denoted by "5TCLWMPR" below.

2) Dataset and Metric for Comparison: We experimented on a public wafer map pattern dataset: WM-811K [14], containing wafer maps collected from real-world manufacturing process in our experimental studies. As shown in Table I, we used 54,355 labeled wafer maps in the dataset, and split them into a training dataset and a testing dataset with a percentage of

TABLE I: WM-811K dataset statistics.

Wafer map patterns	Training	Testing
None	33051	3679
Center	3113	349
Donut	372	37
Edge-Loc	2150	267
Edge-Ring	7735	819
Loc	1458	162
Random	546	63
Scratch	446	54
Near-full	49	5
Total	48920	5435

90% and 10%, respectively. In order to perform the proposed semi-supervised contrastive learning framework, all the 48920 wafer maps in the training dataset are used to construct the unlabeled dataset $\mathcal{D}_U$. Only a small portion $p_d\%$ of the training dataset is collected to build the labeled dataset $\mathcal{D}_L$, containing both wafer maps and the corresponding patterns, which is applied for both the proposed semi-supervised learning method and the supervised learning methods for comparison. A wide range of labeled data percentage $p_d\%$ is experimented as shown in Table III.

Note that the number of samples for each wafer map pattern is highly imbalanced in Table I. For example, the majority pattern, "None" pattern (wafers without any failure pattern), constitutes 67.6% of the whole dataset. Therefore, a traditional accuracy metric doesn't give much information about the wafer map pattern recognizer quality. Even a recognizer predicting all wafer maps as "None" patterns gives an accuracy of 67.6%, which obviously exaggerates the recognizer performance. Instead, we choose a balanced accuracy metric defined as follows.

$$BAC = \frac{\sum_{i=1}^{N} w_i \mathbb{1}\left[\hat{y}_i = y_i\right]}{\sum_{i=1}^{N} w_i}, \tag{9}$$

where y_i and $\hat{y}_i$ is the true data label and the prediction result for a wafer map $\mathbf{x}_i$, respectively. The sample weight w_i balances the imbalance among all the pattern types using $w_i = 1/N_{y_i}$, where N_{y_i} is the number of samples with the same pattern y_i in the dataset. Note that here a trivial recognizer (predicting "None" for all inputs) can only give a balanced accuracy of 11.1% $(= 1/9)$, more fairly presenting the recognizer performance. Furthermore, for the wafer map pattern recognition, a higher balanced accuracy indicates the recognizer is more likely to capture systematic manufacturing failure patterns, which is more meaningful in the semiconductor fabrication.

3) Detailed Hyperparameter Settings: For the SVM, we followed [1] extracting features using Radon-based transform and geometric-based statistics, and trained the SVM recognizer with a radial basis function (RBF) kernel.

For the neural network architecture used in the CNN and the contrasive learning methods, in order to make a fair comparison, the same CNN architecture is used for both the supervised CNN training and the semi-supervised contrastive learning, although different neural network architectures are used in [3]–[6]. In particular, the composition of the encoder $v\left(\cdot;\boldsymbol{\theta}_e\right)$ and the supervised head $h_s\left(\cdot;\boldsymbol{\theta}_s\right)$ used in the contrastive

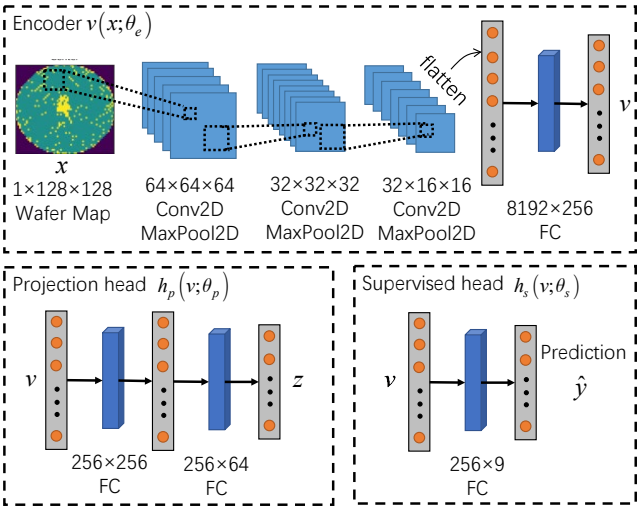


Fig. 9: Neural networks architectures for semi-supervised contrastive learning framework. Note that the supervised CNN uses the same architecture as $f\left(x;\theta_e,\theta_s\right) = h_s\left(v\left(x;\theta_e\right);\theta_s\right)$. learning forms the CNN architecture $f\left(\cdot;\boldsymbol{\theta}_e,\boldsymbol{\theta}_s\right)$ used in a supervised setting. The detailed neural network architectures for the encoder, the projection head, and the supervised head are given in Fig. 9. We resize all the wafer maps to a size of 128×128, and use a vector space with a dimensionality of 256 for the internal representation space $\mathcal{V}$. The entire architectures are implemented using PyTorch v1.8.0 [15] on a workstation with an AMD Ryzen Threadripper 3970X 32-Core processor and an NVIDIA GeForce RTX 3090 GPU.

We experimented two different batch sizes for the semi-supervised contrastive learning $B_U = 256$ and $B_U = 512$ (denoted as "SB" and "LB" in Table III). The training of the encoder uses a maximum epochs of $T_U = 100$ with an early stopping mechanism [16]. The temperature coefficient is set to be 0.1 here. For the supervised head training, another $T_L = 20$ epochs are assigned for finetuning the supervised head with a batch size of $B_L = 64$. Note that in order to alleviate the imbalance issue as shown in Table I, we adopt a weighted batch sampler to balance the occurrence probability of each wafer map pattern during the supervised head training. The same setting is applied to the supervised CNN training with longer epochs $T_L = 100$ to train the entire networks. Moreover, to give more favors to SVM [1] over BAC metric, similar consideration for balancing weighting among different patterns is applied (denoted as "Weighted SVM" in Table III). All the neural network optimization is conducted via an Adam optimizer [17] with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-4}.

B. Evaluating the Performance of Transformations in 4TCLWMPR

In order to justify the usage of the four domain-specific transformations used in 4TCLWMPR, a comprehensive study of the four adopted transformations is performed in Table II. We consider all variants of the possible transformation combinations, with each transformation enabled or not. In total, 24 $(= 2\times2\times3\times2)$ possible transformation combinations are considered (2 from random horizontal flip, 2 from random

TABLE II: Averaged balanced accuracy performance from 24 variants of different transformation combinations in 4TCLWMPR.

Category	Enabled	Labeled data percentage		
		5%	10%	20%
Flip	No	80.71%	80.61%	82.53%
	Yes	**80.90%**	**81.42%**	**83.02%**
Crop	No	80.02%	79.93%	80.99%
	Yes	**81.59%**	**82.09%**	**84.56%**
Rotation	No	80.60%	81.05%	82.17%
	Continuous	80.68%	80.59%	**83.09%**
	Discrete	**81.14%**	**81.40%**	83.07%
Noise	No	80.26%	80.24%	81.73%
	Yes	**81.35%**	**81.79%**	**83.82%**

resized cropping, 3 from random wafer rotation, and 2 from random die noise,). Each row in Table II gives the averaged balanced accuracy of the transformation combinations with the enabled transformation category. For instance, the row of random "Discrete" wafer rotation averages the balanced accuracy from the corresponding 8 transformation combinations.

As shown in Table II, three of selected transformations: random horizontal flipping, random resized cropping, and random die noise, always improve the resulting prediction accuracy for all the labeled data percentages considered. For the random wafer rotation, although the "Discrete" variant's performance is slightly lower than the "Continuous" one for a labeled data percentage of 20%, a relatively large drop 0.46% of "Continuous" variant can be observed at a labeled data percentage of 10% compared to disabling wafer rotation. With that, we include the "Discrete" version of random wafer rotation in the final selected transformations.

C. Semi-supervised Learning Performance

The wafer map pattern recognition performance over the WM-811K dataset for the semi-supervised learning method is given in Table III. The first three rows are the balanced accuracy metric for the three supervised methods: SVM [1], Weighted SVM, and the CNN architecture. The next 6 rows are the performance for the baseline SimCLR and the proposed semi-supervised contrastive learning variants with different transformation combinations and batch sizes. As we can see from the table, the recognition balanced accuracy for the supervised methods is relatively low. For the traditional SVM classifier in [1], the recognition balanced accuracy is even lower than 50%. Even with balanced sample weights to train the SVM, the recognizer can be hardly to be used in a real application to identify wafer map failure patterns, although the averaged balanced accuracy of weighted SVM gets improved by 4.56% compared to the vanilla SVM. With the help of modern neural network architectures like CNN, the recognition power gets greatly enhanced by another increase of 19.89% in terms of the average balanced accuracy.

Thanks to the good representation learnt by the contrastive learning using an unlabeled dataset, the proposed semi-supervised learning framework significantly improves the recognition performance with a boost around 7% for the balanced accuracy on average. For instance, we can observe a recognition accuracy boost of 9.77% using the proposed contrastive learning with the selected transformations, compared to CNN, even for a small labeled data percentage like 1%. These significant wafer map pattern recognition performance improvements can be attributed to two aspects: 1) Good representation for wafer maps is well extracted by comparing different transformations of wafer maps to learn the similarity among samples; 2) The unlabeled data is fully utilized, which cannot be learnt by supervised methods, greatly reducing the manual labor to label all the wafer map patterns.

The effect of different transformation combinations is also studied for the semi-supervised contrastive learning framework. It is observed from Table III that there still exist certain performance gaps between different transformation combinations, although even the baseline SimCLR transformations' performance already surpasses the supervised CNN a lot. With the domain-specific transformations (4TCLWMPR), the average balanced accuracy over all labeled data percentages get improved by 1.23% for small batch size and 1.65% for large batch size. Another 0.76% boost can be observed when applying the proposed *rotation-twist* transformations (5TCLWMPR). With the additional proposed *rotation-twist* transformation (5TCLWMPR), there are 5 cases outperforming all other methods using a small batch size for all the 7 experimented labeled data percentages, and 6 out of 7 using a large batch size. Such domain-specific transformations suggest the similarity among wafer maps are well captured by these selected transformations, enhancing the representation learnability for the wafer map pattern recognition application. The proposed *rotation-twist* transformation further refines the representation learning by identifying similar samples with distinct detailed failure pattern shapes.

As suggested in [9], a large batch size is beneficial for the contrastive learning, as more dissimilar views of samples can be used in (4) to identify the dissimilarity among samples. As shown in Table III, with a large batch size, the balanced recognition accuracy can be slightly improved on average, however, we still suggest a case-by-case study for the batch size to be used in contrastive learning when applying different labeled data percentages.

Note that we also experimented another variant of contrastive learning which also finetunes the encoder during the second phase of the semi-supervised learning. However, a similar performance is observed as the one for supervised CNN, implying that the finetuned encoder forgets the representation learnt during the unsupervised contrastive learning and degrades to a simple supervised learning.

D. Rotation-Twist as Data Augmentation

To boost the recognition power for a supervised learning model, one common strategy is to augment a small dataset by adding more samples transformed from the original dataset with the corresponding data labels. To validate the effectiveness of our proposed *rotation-twist* transformation, we experiment the same CNN architecture, and train them in a supervised manner with each sample randomly transformed by the proposed *rotation-twist* transformation with an order

TABLE III: Balanced accuracy comparison for WM-811K between supervised learning and semi-supervised learning methods.

Learning Type	Method	Labeled data percentage							Avg.
		1%	5%	8%	10%	20%	30%	50%	
Supervised	SVM	47.27%	41.57%	47.79%	45.88%	48.66%	N/A	46.99%	46.36%
	Weighted SVM	48.07%	46.99%	51.70%	54.72%	55.40%	N/A	48.76%	50.94%
	CNN	64.87%	66.10%	71.41%	66.74%	77.90%	74.62%	74.20%	70.83%
Contrastive Learning (SB)	SimCLR	71.20%	71.11%	71.15%	72.81%	81.13%	79.22%	82.42%	75.58%
	Proposed 4TCLWMPR	**74.64%**	70.16%	73.38%	73.18%	**83.05%**	79.94%	83.32%	76.81%
	Proposed 5TCLWMPR	72.81%	**74.65%**	**76.48%**	**75.48%**	79.75%	**80.26%**	**83.47%**	**77.56%**
Contrastive Learning (LB)	SimCLR	72.35%	72.05%	75.24%	70.39%	78.30%	80.10%	79.91%	75.48%
	Proposed 4TCLWMPR	72.42%	73.90%	76.34%	**75.08%**	80.23%	81.75%	80.21%	77.13%
	Proposed 5TCLWMPR	**73.27%**	**75.54%**	**77.52%**	73.70%	**82.19%**	**82.39%**	**80.72%**	**77.90%**

TABLE IV: Performance boost of *rotation-twist* transformation as data augmentation in a supervised learning setting.

Labeled data percentage	CNN	CNN+RoTwist	Improvement
1%	64.87%	68.19%	+3.32%
5%	66.10%	73.25%	+7.15%
8%	71.41%	73.10%	+1.69%
10%	66.74%	73.60%	+6.86%
20%	77.90%	81.39%	+3.49%
30%	74.62%	80.62%	+6.00%
50%	74.20%	78.06%	+3.86%
Avg.	70.83%	75.46%	+4.62%

of $m = 3$. Table IV gives the resulting balanced accuracy performance with *rotation-twist* data augmentation at different labeled data percentages. Compared to the vanilla CNN, a great accuracy improvement can be observed with 4.62% on average, indicating the effectiveness of the proposed transformation, which well extracts the similarity among wafer maps.

Compared to the semi-supervised contrastive learning results as shown in Table III, the proposed contrastive learning recognition still outperforms data-augmented supervised CNN for all the experimented labeled data percentages, demonstrating that the proposed semi-supervised contrastive learning can efficiently learn good representations for wafer maps to build a robust wafer map pattern recognizer.

VII. Conclusion

In this paper, we proposed a semi-supervised contrastive learning framework for wafer map pattern recognition. Contrastive learning is adopted to learn good representation in an unsupervised manner via comparing different views of wafer maps generated by a set of selected domain-specific transformations. In addition, a novel *rotation-twist* transformation is proposed to change the detailed shape of wafer maps while maintaining the original patterns. Our experimental results demonstrate the effectiveness of the semi-supervised contrastive learning over the supervised learning methods, and present the performance boost for the proposed domain-specific transformations.

Acknowledgement

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 1956313 and Semiconductor Research Corporation (SRC) Task No. 2810.031 through UT Dallas' Texas Analog Center of Excellence. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF and SRC.

References

[1] M.-J. Wu, J.-S. R. Jang, and Jui-Long Chen, "Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets," *IEEE Transactions on Semiconductor Manufacturing*, vol. 28, no. 1, pp. 1–12, Feb. 2015.

[2] M. Fan, Q. Wang, and B. van der Waal, "Wafer defect patterns recognition based on OPTICS and multi-label classification," in *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, Oct. 2016, pp. 912–915.

[3] M. B. Alawieh, D. Boning, and D. Z. Pan, "Wafer Map Defect Patterns Classification using Deep Selective Learning," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, Jul. 2020, pp. 1–6.

[4] R. di Bella, D. Carrera, B. Rossi, P. Fragneto, and G. Boracchi, "Wafer Defect Map Classification Using Sparse Convolutional Networks," in *Image Analysis and Processing – ICIAP 2019*, Cham, 2019, pp. 125–136.

[5] D.-Y. Du and Z. Shi, "A Wafer Map Defect Pattern Classification Model Based on Deep Convolutional Neural Network," in *2020 IEEE 15th International Conference on Solid-State Integrated Circuit Technology (ICSICT)*, Nov. 2020, pp. 1–3.

[6] T.-H. Tsai and Y.-C. Lee, "A Light-Weight Neural Network for Wafer Map Classification Based on Data Augmentation," *IEEE Transactions on Semiconductor Manufacturing*, vol. 33, no. 4, pp. 663–672, Nov. 2020.

[7] M. Nero, C. Shan, L.-C. Wang, and N. Sumikawa, "Concept Recognition in Production Yield Data Analytics," in *2018 IEEE International Test Conference (ITC)*, Oct. 2018, pp. 1–10.

[8] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, "Big Self-Supervised Models are Strong Semi-Supervised Learners," *arXiv:2006.10029 [cs, stat]*, Oct. 2020, arXiv: 2006.10029.

[9] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," *arXiv:2002.05709 [cs, stat]*, Jun. 2020, arXiv: 2002.05709.

[10] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive Representation Learning: A Framework and Review," *IEEE Access*, vol. 8, pp. 193 907–193 934, 2020.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[12] H. Kahng and S. B. Kim, "Self-supervised representation learning for wafer bin map defect pattern classification," *IEEE Transactions on Semiconductor Manufacturing*, vol. 34, no. 1, pp. 74–86, 2021.

[13] S. Filip, A. Javeed, and L. N. Trefethen, "Smooth random functions, random odes, and gaussian processes," *SIAM Rev.*, vol. 61, no. 1, p. 185–205, Jan. 2019.

[14] Q. Yi. (2018) WM-811K wafer map. [Online]. Available: https://www.kaggle.com/qingyi/wm811k-wafer-map

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.

[16] A. Lodwich, Y. Rangoni, and T. Breuel, "Evaluation of robustness and performance of early stopping rules with multi layer perceptrons," in *2009 International Joint Conference on Neural Networks*, 2009, pp. 1877–1884.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.