Self-Supervised Representation Learning for Wafer Bin Map Defect Pattern Classification

Hyungu Kahng and Seoung Bum Kim, *Member, IEEE*

***Abstract*—Automatic identification of defect patterns in wafer bin maps (WBMs) stands as a challenging problem for the semiconductor manufacturing industry. Deep convolutional neural networks have recently shown decent progress in learning spatial patterns in WBMs, but only at the expense of explicit manual supervision. Unfortunately, a clean set of labeled WBM samples is often limited in both size and quality, especially during rapid process development or early production stages. In this study, we propose a self-supervised learning framework that makes the most out of unlabeled data to learn beforehand rich visual representations for data-efficient WBM defect pattern classification. After self-supervised pre-training based on noise-contrastive estimation, the network is fine-tuned on the available labeled data to classify WBM defect patterns. We argue that self-supervised pre-training with a vast amount of unlabeled data substantially improves classification performance when labels are scarce. We demonstrate the effectiveness of our work on a real-world public WBM dataset, WM-811K. The code is available at https://github.com/hgkahng/WaPIRL.**

***Index Terms*—Wafer bin map defect pattern classification, semiconductor manufacturing, convolutional neural networks, unsupervised learning, self-supervised learning, contrastive learning.**

I. Introduction

SEMICONDUCTOR device fabrication is a complex manufacturing process involving hundreds of serialized steps designated to assemble integrated circuit (IC) chips on silicon wafers. Driven by the ever-growing demands of the global market, semiconductor manufacturers are pushing to increase productivity by shrinking individual node sizes and by integrating more chips on a single wafer. This greatly complicates the whole fabrication process, resulting in frequent inline defects that cause circuit failures and hinder overall yield. With the increased complexity of modern production environments, the industry ardently demands sophisticated techniques for wafer quality assessment in order to reduce production costs in the effort to remain competitive.

In particular, great interest lies in analyzing spatial defect patterns in wafer bin maps (WBMs), which carry valuable information for pinpointing process failures and their associated root causes. After wafer fabrication and before packaging, every wafer is subjected to a series of multiple electrical tests to verify functional defects in IC chips. A WBM is basically a two-dimensional grid portraying these binary test results based on their physical locations. It is important to note that defect patterns in WBMs are often systematic, i.e., they arise from specific process failures that occur during fabrication. Therefore, correctly classifying defect patterns is closely tied to identifying their root causes and providing proper treatment.

The conventional practice of relying on experienced engineers to analyze WBMs with the naked eye is no longer a valid option and requires a breakthrough replacement [10]. Because modern fabs produce several thousands of wafers every week, it is practically impossible for engineers to manually go through every WBM and categorize them into one of the predefined defect types. It is often the case that engineers' decisions are biased depending on their levels of experience or working conditions. Furthermore, previously unknown defect patterns may emerge early during the process development of new production stages [7]. An almost measureless volume of WBM data streaming amidst large-scale production remains untreated, even though it may embody worthwhile information for root cause identification and thus yield improvements. Automated systems that can exploit this vast amount of unexamined data are sure to benefit the analysis of WBM defect patterns.

Undera setting where labeled data is scarce but unlabeled data is abundant, unsupervised learning serves as a promising strategy for the development of entirely automated classifiers. In the field of computer vision, a recent line of research [12], [13], [19], [20], [21], [23], [26], [27], [28], [29], [31], [32], [35] falling under the umbrella of self-supervised learning has shown the potential to learn good feature representations solely from large-scale unlabeled images. By pre-training convolutional neural networks (CNNs) to learn an auxiliary task that does not require costly human annotations, this new learning paradigm seeks to improve the generalization performance of classification models when labels are limited. Our work builds on the intuition that the WBM defect pattern classification task can also benefit from transferring representations learned with self-supervision from unlabeled WBM data.

Manuscript received September 9, 2020; revised October 9, 2020 and October 31, 2020; accepted November 11, 2020. Date of publication November 16, 2020; date of current version February 3, 2021. This work was supported in part by the Korea Institute for Advancement of Technology (KIAT) Grant funded by the Korean Government (MOTIE) under Grant P0008691 (Competency Development Program for Industry Specialists), and in part by the National Research Foundation of Korea Grant funded by the Korean Government (MSIT) under Grant NRF-2019R1A4A1024732. *(Corresponding author: Seoung Bum Kim.)*

The authors are with the School of Industrial Management Engineering, Korea University, Seoul 02841, South Korea (e-mail: hgkahng@korea.ac.kr; sbkim1@korea.ac.kr).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSM.2020.3038165.

Digital Object Identifier 10.1109/TSM.2020.3038165

In this study, we propose a neural network pre-training method based on self-supervised learning to improve the performance of WBM defect pattern classification when only a subset of the samples have corresponding labels. We hypothesize that the representations learned from self-supervision are broadly transferable and can serve as proper initial parameters for enhancing the performance of downstream classification models. Specifically, a CNN encoder is pre-trained to learn semantic features by mapping contextually similar WBMs to nearby points in an embedding space. The mapping is achieved by minimizing the normalized distance between features extracted from an original WBM and its augmented correspondents. After pre-training, the CNN encoder is fine-tuned to classify WBMs on available labeled data. We find that not only does our method improve classification performance with a limited set of labels, but it also consistently outperforms fully supervised baselines trained with larger proportions of labeled data. To the best of our knowledge, exploiting the power of self-supervised learning has not been considered in the context of WBM defect pattern classification.

The remainder of the paper is organized as follows. Section II gives a brief review of the studies on WBM defect pattern classification and recent advances in self-supervised learning. Section III illustrates the details of the proposed methodology. Experimental results and findings are provided in Section IV. Finally, Section V contains concluding remarks with directions for future research.

II. Related Work

The literature contains numerous studies for classifying WBM defect patterns. We limit our scope to the relatively new body of research based on statistical learning, machine learning, and deep learning. We also provide a broad review of the most prominent research that has spurred the rapid progress of self-supervised learning for visual recognition tasks.

A. Wafer Bin Map Defect Pattern Classification

Wang *et al.* [1] proposed a hybrid clustering method with spatial filters based on Gaussian expectation-maximization and a spherical shell algorithm to identify defect patterns. Jeong *et al.* [2] observed the presence of spatial autocorrelations in defective WBMs and applied spatial correlograms for feature extraction and dynamic warping for clustering. Li and Huang [3] proposed training self-organized maps to discover representative defect patterns and used the assigned clusters as hard labels for training SVMs. For WBM similarity searching, Liao *et al.* [6] proposed generating synthetic samples with a morphology-based SVM. A regularized singular value decomposition method was developed by Kim *et al.* [15] to distinguish between single and non-single bit failure maps. Adly *et al.* [11] introduced general-regression-based consensus learning with bootstrap aggregation. Yu and Lu [16] proposed a joint local and nonlocal linear discriminant analysis method to extract features from WBMs. Saqlain *et al.* [33] proposed training a soft voting ensemble classifier based on a combination of density, randon, and geometry-based features. The methods above mostly insist on rigorous feature engineering followed by a statistical or machine learning model.

On the other hand, neural network-based approaches have been explored for classifying WBM defect patterns by taking advantage of the recent advent of deep learning. Wang and Chen [34] trained a multilayer perceptron on features extracted with a set of hand-designed spatial filters reflecting the rotation-invariant property of WBMs. Nakazawa and Kulkarni [25] applied CNNs to classify defect patterns based on a synthetic WBM dataset, and Kyeong and Kim [24] applied CNNs to a WBM dataset with mixed-type defect patterns. Saqlain *et al.* [39] demonstrated the effectiveness of WBM oversampling based on data augmentation to cope with class imbalance when training CNNs. Kim *et al.* [30] proposed using an unsupervised embedding of WBMs and a CNN model to identify different defect types. Although these methods appear to yield high test accuracies in their individual settings, they are limited to supervised training with a readily available set of clean labels. Recently, Kong and Ni [37] applied ladder networks to incorporate unlabeled data under a semi-supervised setting; however, the dataset used in their experiments only consisted of 638 WBMs, which is somewhat insufficient to demonstrate the model's scalability.

B. Self-Supervised Learning

In the context of deep learning, self-supervised learning aims to learn robust feature representations directly from the input data in the absence of sufficient class labels. The major challenge of self-supervised learning is to design auxiliary pretext tasks in a way that solving them encourages a neural network to learn fruitful representations. By exploiting high-level semantics of the input data, this new learning paradigm has reached out to various data domains, including computer vision, signal processing, natural language processing, and robotics. We restrict our interest to the image domain. One line of work in that domain focuses on predictive approaches, where one part of the input image is left out to be predicted from the other parts of the image. Research in this direction includes but is not limited to predicting relative locations of patches cropped from a single image [12], matching exemplars [13], inpainting missing patches based on the remaining context [20], solving jigsaw puzzles [19], image colorization from grayscale to RGB [21], and predicting image orientations [23]. However, it has been determined that the learned representations are often covariant to the design of the pretext task, leading to poor generalization performance on downstream tasks such as image classification and object detection [32].

More recently, a new wave of methods has built on the concept of contrastive representation learning [26], [27], [28], [29], [32], [35], [36] to compensate for the drawbacks induced by heuristically-designed pretext tasks. In brief, the core idea behind these approaches is that different versions of an image obtainable through various data augmentations, should map to a similar embedding which distinguishes them from other contextually different images. Surprisingly,

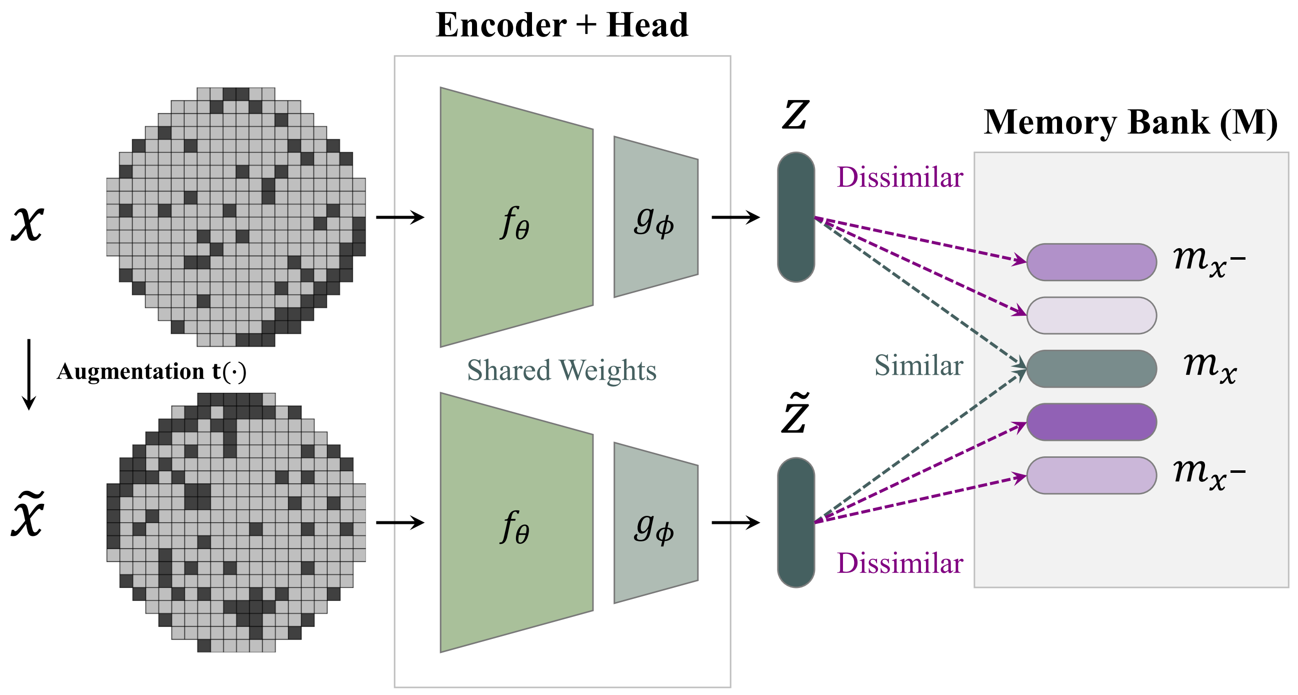


Fig. 1. A schematic overview of our proposed pre-training method, WaPIRL.

contrastive learning-based methods have pushed the limits of self-supervised representations to match those obtained from supervised learning with many labels.

Among the many recent methods, we adapt the pretext invariant representation learning (PIRL) framework proposed by Misra and van der Maaten [32] for WBM defect pattern classification. We demonstrate that our method is capable of learning pretext-invariant features from an extensive set of WBMs without any class labels.

III. Methodology

Our self-supervised learning framework for WBM defect pattern classification mainly consists of two stages: 1) the self-supervised pre-training stage and 2) the supervised fine-tuning stage. In the self-supervised pre-training stage, we train a CNN encoder that captures high-level semantic features in WBMs without any supervisory signals. In the supervised fine-tuning stage, we transfer the pre-trained weights and train a CNN classifier to predict WBM defect patterns with available class labels. This section describes the proposed framework in detail.

A. Self-Supervised Pre-Training

The goal of the self-supervised pre-training stage is to learn rich feature representations that embody the semantic similarities across WBMs. We achieve this by minimizing a contrastive loss function that encourages features extracted from an original, non-augmented WBM and its augmented correspondents to cluster together. Fig. 1 provides a schematic overview of our methodology, wafer-oriented pretext-invariant representation learning (WaPIRL, pronounced 'Way Pearl'), for self-supervised pre-training on WBM data. The proposed learning framework mainly comprises five components: a data augmentation operator $t(\cdot)$, an encoder network $f_\theta(\cdot)$, a projection head $g_\phi(\cdot)$, a memory bank $\mathcal{M}$, and a self-supervised contrastive loss function $\mathcal{L}_{SSCL}$. Note that the subscripts in $f_\theta(\cdot)$ and $g_\phi(\cdot)$ indicate that the networks are trainable with their respective parameters θ and ϕ. Denote a training dataset of N_u unlabeled WBMs as $\mathcal{D} = \{x_i\}_{i=1}^{N_u}$ consisting of WBMs defined as $x \in \mathbb{R}^{H \times W}$, where H and W refer to the height and width. The remainder of this article will be developed on these notations.

The data augmentation operator $t(\cdot)$ is employed to produce positive pairs of WBMs that are used for self-supervised contrastive learning. Given an original, non-augmented WBM, x_i, the data augmentation operator transforms it into a randomly augmented version, $\tilde{x}_i = t(x_i)$. We can regard this augmentation process as generating a new corrupted view of each WBM while preserving its high-level context (i.e., class-specific patterns). To understand how WaPIRL performs when trained with different data augmentations, we experimented with five different choices: *cropping*, *cutout*, *noise addition*, *rotation*, and *shifting*. Visualizations of these augmentations are provided in Fig. 2. *Cropping* involves randomly selecting a rectangular region and resizing it to the original resolution. The crop scale factor is randomly chosen from [0.5, 1.0]. For *cutout*, a maximum of four square regions were randomly selected and removed; the corresponding values were replaced with zeros. For *noise addition*, we added random noise to WBMs using a Bernoulli distribution with a probability of 0.05. *Rotation* was performed with an angle ranging between $[0°, 360°]$ to obtain WBMs of different orientations. Finally, *shifting* randomly translates the image in both the horizontal and vertical directions, with a shifting factor randomly chosen from $[-0.25, 0.25]$.

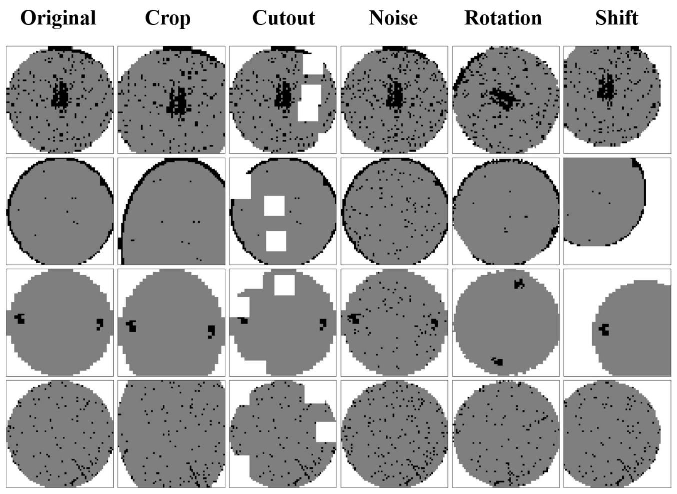


Fig. 2. Five different augmentations used in our work.

The encoder network $f_\theta(\cdot)$, which is a CNN parameterized by θ, maps both x_i and its augmented version $\tilde{x}_i$ to fixed-size representation vectors, $v_i = f_\theta(x_i) \in \mathbb{R}^{D_f}$ and $\tilde{v}_i = f_\theta(\tilde{x}_i) \in \mathbb{R}^{D_f}$. To investigate the relationship between model capacity and the quality of self-supervised representations, we conducted experiments with four popular CNN architectures with varying depths: AlexNet [5], VGG16 [8], ResNet-18, and ResNet-50 [17]. Several recent studies engaged in a variety of visual recognition tasks have demonstrated that self-supervised pre-training delivers larger improvements with highly capacitated models, and we intend to verify whether this phenomenon holds for WBMs as well. We believe that the four architectures suffice as suitable baselines to demonstrate the effect of self-supervised pre-training [31]. For AlexNet and VGG16, a global average pooling layer is substituted for the fully-connected layers to obtain a D_f-dimensional representation vector. For ResNet-18 and ResNet-50 which already have a global average pooling layer after the last convolutional layer, we need only remove the last fully connected layer. With such adjustments, D_f is determined by the number of feature maps obtained from the last convolutional layer. Architectural details are provided in the Appendix (Tables IV, V, VI, and VII). Meanwhile, given that our proposed method is a generic learning framework, we note that the encoder $f_\theta(\cdot)$ can be easily replaced with any standard or non-standard CNN architecture. The only prerequisite is that the final output layer produces a fixed-size representation vector.

The projection head $g_\phi(\cdot)$ independently receives D_f-dimensional representation vectors v_i and $\tilde{v}_i$ extracted by the CNN encoder network $f_\theta(\cdot)$, and maps them to D_g-dimensional embedding vectors $z_i = g_\phi(v_i) \in \mathbb{R}^{D_g}$ and $\tilde{z}_i = g_\phi(\tilde{v}_i) \in \mathbb{R}^{D_g}$. In our experiments, we follow the work by [32] and use a single linear (fully-connected) layer for $g_\phi(\cdot)$ with a fixed output size D_g of 128. The scale of the embedding vectors are l_2-normalized such that $|z_i| = |\tilde{z}_i| = 1$, constraining the representations to lie on a unit hypersphere. Normalization enables the use of a simple inner product operation to measure the similarities between a pair of sampled WBM embedding vectors in the projection space. Note that the projection head is only necessary during the self-supervised pre-training stage to produce the embedding vectors used for computing the contrastive loss function. In the supervised fine-tuning stage, the projection head is replaced with a linear softmax layer for the purpose of downstream classification. Further details are provided later in Section III-B.

The memory bank $\mathcal{M}$ maintains an exponential moving average of the embedding vectors $z_i = g_\phi(f_\theta(x_i))$ for every unlabeled training sample x_i, where $i \in \{1, 2, \ldots, N_u\}$. The role of the memory bank is to mitigate the computational burden of negative sampling, in the context of noise contrastive estimation (NCE) [4], [22], [27], [32]. The update rule for the memory of the i^{th} sample, $m_i \in \mathbb{R}^{D_g}$, is defined as follows:

$$m_i \leftarrow \gamma m_i + (1-\gamma) z_i, \tag{1}$$

where $\gamma \in [0, 1]$. Following the work by [32], we use the default value of $\gamma = 0.5$ to assign equal weights to the past and current states during updates. At the beginning of training, the memory values for every training sample are initialized by performing a forward pass through the CNN encoder and projection head. Given a pair of positive embedding vectors $(z_i, \tilde{z}_i)$, our objective is to maximize their similarities by NCE, which models the distribution $p(z_i, \tilde{z}_i)$ as the probability of the binary event that the two positive representation vectors $(z_i, \tilde{z}_i)$ arise from the same distribution. For the i^{th} sample, NCE in its most general form can be formulated as follows:

$$p(z_i, \tilde{z}_i) = \frac{\exp\left(\frac{s(z_i,\tilde{z}_i)}{\tau}\right)}{\exp\left(\frac{s(z_i,\tilde{z}_i)}{\tau}\right) + \sum_{j \in \mathcal{D}-\{i\}} \exp\left(\frac{s(z_i,z_j)}{\tau}\right)}. \tag{2}$$

$s(\cdot,\cdot)$ is a normalized similarity function (i.e., inner dot product) ranging $[-1, 1]$, and τ is a temperature parameter used to scale the similarities. Maximizing the likelihood of (2) over the training data maximizes the similarities between positive representations z_i and $\tilde{z}_i$ originating from an identical WBM, while minimizing the similarities between negative representations z_i and $\{z_j\}_{-i}^{N_u} = \{z_j | j = 1, 2, \ldots, i-1, i+1, \ldots, N_u\}$ originating from different WBMs. However, calculating the summation term in the denominator of (2) is computationally expensive because it is defined over all negative training examples. As an alternative, negative sampling approximates (2) by randomly sampling K ($< N_u$) negative examples. Thus, NCE with negative sampling defined for the i^{th} sample is formulated as

$$p(z_i, \tilde{z}_i) = \frac{\exp\left(\frac{s(z_i,\tilde{z}_i)}{\tau}\right)}{\exp\left(\frac{s(z_i,\tilde{z}_i)}{\tau}\right) + \sum_{j \in \mathcal{J}_{-i}^{K}} \exp\left(\frac{s(z_i,z_j)}{\tau}\right)}, \tag{3}$$

where $\mathcal{J}_{-i}^{K} \subseteq \mathcal{J} - \{i\}$ denotes an index set for the K negative examples drawn from the unlabeled training data $\mathcal{D}$ while excluding index i. Unfortunately, computing (3) still requires K forward passes through the network to obtain a set of K embedding vectors $\{z_1, z_2, \ldots, z_{i-1}, z_{i+1}, \ldots, z_K\}$. Instead, incorporating a memory bank alleviates the heavy computation of forward passes by using an online approximation $m_j \approxeq z_j$, thus allowing (3) to be reformulated as follows:

$$p(z_i, m_i) = \frac{\exp\left(\frac{s(z_i,m_i)}{\tau}\right)}{\exp\left(\frac{s(z_i,m_i)}{\tau}\right) + \sum_{j \in \mathcal{J}_{-i}^{K}} \exp\left(\frac{s(z_i,m_j)}{\tau}\right)}. \tag{4}$$

With the modification from (3) to (4), the K negative examples are now efficiently sampled from the memory bank. The computational complexity of the negative sampling term is reduced to $\mathcal{O}(\log K)$, which is equivalent to sampling from a finite discrete probability distribution. Additionally, we substitute $\tilde{z}_i$ in (3) with its memory representation m_i in (4). The loss function for NCE with negative sampling along with a memory bank adds up to the negative log-likelihood of (4) as:

$$\mathcal{L}_{NCE}(z_i, m_i) = -\log(p(z_i, m_i)). \quad (5)$$

Ultimately, the self-supervised contrastive loss function $\mathcal{L}_{SSCL}$ takes the form of a convex combination of two NCE loss functions from (5) as:

$$\mathcal{L}_{SSCL} = \frac{1}{N_u}\sum_{i=1}^{N_u} \lambda\mathcal{L}_{NCE}(z_i, m_i) + (1-\lambda)\mathcal{L}_{NCE}(\tilde{z}_i, m_i), \quad (6)$$

where $\lambda \in [0, 1]$. The hyperparameter λ balances the effect of the two loss terms. The first term encourages the embedding vector z_i extracted from an original WBM to be similar to its memory representation m_i, while the second term encourages the embedding vector $\tilde{z}_i$ extracted from an augmented WBM to be similar to its memory representation m_i. From a triplet loss perspective, the memory representation m_i plays the role of an anchor point, pulling z_i and $\tilde{z}_i$ closer while pushing away the K negative examples. We simply set $\lambda = 0.5$, which has been reported in [32] to give the best performance.

B. Supervised Fine-Tuning

The supervised fine-tuning stage aims to train a CNN classifier by taking advantage of the pre-trained representations learned beforehand by self-supervised contrastive learning. First, we do away with the final projection head $g_\phi(\cdot)$ and stack a linear layer $h_\omega(\cdot)$, followed by a softmax activation function to normalize the output logits into probabilities. A dropout layer is applied with a fixed probability of 0.5 before the linear layer to prevent overfitting [9]. In addition, we apply label smoothing to the one-hot encoded labels to mitigate the damaging effect of noisy labels and to enhance model calibrations [14]. Given a labeled training dataset of N_l WBMs with C distinct defect pattern classes, our loss function for supervised fine-tuning, $\mathcal{L}_{SL}$, is a categorical cross-entropy function with label smoothing formulated as follows:

$$\mathcal{L}_{SL} = -\sum_{i=1}^{N_l}\sum_{c=1}^{C} \tilde{y}_{i,c} \cdot \log \hat{y}_{i,c}. \quad (7)$$

Herein, $\tilde{y}_{i,c} = (1-\alpha)\cdot y_{i,c} + \frac{\alpha}{C-1}\cdot(1-y_{i,c})$ is a soft target value ranging [0, 1], given a one-hot encoded vector $y_i \in \mathbb{R}^C$ and a smoothing factor $\alpha \in [0, 1]$. The model prediction $\hat{y}_{i,c} \in [0, 1]$ is the normalized probability of the i^{th} WBM sample belonging to class c. Using $\alpha = 0.0$ reduces (7) to the standard cross entropy loss with hard targets. In all our experiments, we follow the default settings stated in the original paper [32] and set $\alpha = 0.1$.

TABLE I
CLASS LABEL DISTRIBUTION OF 172,950 LABELED SAMPLES IN WM-811K

Class label	Sample counts	Proportion (%)
Center	4,294	2.48
Donut	555	0.32
Edge-Loc	5,189	3.00
Edge-Ring	9,680	5.59
Loc	3,593	2.07
Random	886	0.51
Scratch	1,193	0.68
Near-Full	149	0.09
None	147,431	85.24

IV. EXPERIMENTAL RESULTS

In this section, we report experimental results conducted on a large-scale WBM dataset to demonstrate the benefits conferred by self-supervised pre-training. First and foremost, we varied the amount of labeled training data used during fine-tuning and compared classification performances against supervised baselines (without pre-training) to illustrate the effectiveness of our approach. We also compared the classification performance of WaPIRL with other previously proposed methods. Moreover, we provide analyses of WaPIRL to show that it learns a relevant feature space characterizing the high-level semantic similarities of WBMs.

A. Dataset

We evaluated the performance of WaPIRL on the publicly accessible WBM dataset, WM-811K [10], which comprises 811457 samples obtained from a real-world fabrication process. Only a subset of 172950 samples are associated with human-annotated labels belonging to one of nine predefined classes, leaving out a massive unlabeled set of 638507 samples. The class label distribution is highly imbalanced; the dominant class accounts for roughly 85% of the whole population. The exact sample counts and proportions are given in Table I. Under such severe class imbalance, supervised learning models are prone to overfit to the majority class during training. Considering the batchwise nature of neural network optimization, we applied batch-level resampling to alleviate the class imbalance problem. Specifically, we calculated sample weights based on inverse class frequencies and dynamically sampled from this multinomial distribution to yield balanced minibatches of samples.

WM-811K assigns a value of one for pass bins, two for fail bins, and zero for null bins indicating locations outside the circular WBM region. Meanwhile, the dataset is composed of WBMs with different spatial resolutions. As training CNNs require the input sizes to be fixed, we resized every WBM to a size of 96 × 96 using nearest neighbor interpolation. Smaller sizes showed a degradation in classification performance, while larger sizes only offered marginal improvements.

B. Training Configurations

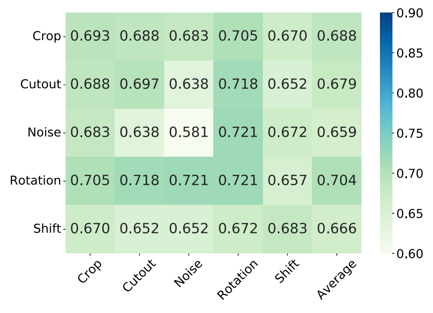
For self-supervised pre-training, we randomly split the unlabeled dataset into two subsets; 80% for model training and 20% for validation. We trained WaPIRL for 100

TABLE II
CLASSIFICATION PERFORMANCES USING ALEXNET, VGG16, RESNET-18, AND RESNET-50 ENCODERS, REPORTED IN TERMS OF MACRO F_1 SCORES. FOR EACH ENCODER ARCHITECTURE AND LABELED DATA SETTING, THE MACRO F_1 SCORE FOR THE BEST PERFORMING AUGMENTATION METHOD IS IN **BOLD**. FOR EACH PROPORTION OF LABELED DATA, SCORES OF THE BEST PERFORMING MODEL ACROSS ARCHITECTURES ARE UNDERLINED (ONE FOR EACH COLUMN)

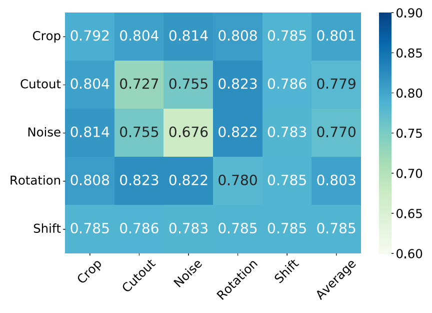
Encoder Architecture	Training Method	Augmentation	Labeled Data					
			1,383 (1%)	6,918 (5%)	13,836 (10%)	34,590 (25%)	69,180 (50%)	138,630 (100%)
AlexNet	Supervised Baseline		0.581	0.667	0.705	0.757	0.793	0.840
	WaPIRL	Crop	0.708	0.783	**0.831**	0.849	0.870	0.885
		Cutout	0.681	0.761	0.797	0.822	0.857	0.887
		Noise	0.622	0.721	0.756	0.807	0.845	0.877
		Rotation	**0.724**	**0.786**	0.827	0.846	0.863	0.864
		Shift	0.694	0.775	0.822	**0.854**	**0.872**	**0.886**
VGG16	Supervised Baseline		0.554	0.681	0.711	0.782	0.846	0.871
	WaPIRL	Crop	0.741	$\underline{\textbf{0.815}}$	$\underline{\textbf{0.839}}$	$\underline{\textbf{0.864}}$	$\underline{\textbf{0.880}}$	$\underline{\textbf{0.897}}$
		Cutout	0.688	0.804	0.824	0.847	0.876	0.889
		Noise	0.585	0.729	0.740	0.807	0.848	0.886
		Rotation	$\underline{\textbf{0.756}}$	0.813	0.828	0.855	0.869	0.870
		Shift	0.739	0.805	0.832	0.860	0.876	0.888
ResNet-18	Supervised Baseline		0.570	0.666	0.716	0.770	0.810	0.858
	WaPIRL	Crop	0.693	**0.792**	0.821	**0.857**	**0.879**	**0.895**
		Cutout	0.697	0.727	0.773	0.838	0.869	0.883
		Noise	0.581	0.676	0.723	0.805	0.857	0.875
		Rotation	**0.721**	0.780	**0.827**	0.846	0.858	0.860
		Shift	0.683	0.785	0.811	0.856	0.876	0.883
ResNet-50	Supervised Baseline		0.558	0.629	0.676	0.785	0.847	0.870
	WaPIRL	Crop	0.684	**0.794**	**0.823**	**0.857**	0.873	**0.892**
		Cutout	0.697	0.737	0.775	0.839	0.852	0.888
		Noise	0.635	0.686	0.739	0.812	0.848	0.880
		Rotation	**0.710**	0.786	0.808	0.840	0.865	0.872
		Shift	0.677	0.771	0.821	0.856	**0.874**	0.890

epochs with a batch size of 256, resulting in a total of 199600 parameter updates. We optimized the parameters with the Stochastic Gradient Descent (SGD) optimizer using a momentum factor of 0.9, a base learning rate of 0.01, and a weight decay factor of 0.001. Starting from the base value, the learning rate was decayed by following a half cosine learning rate schedule [18], until it reached a minimum value of 0.0001. Both the base learning rate and weight decay factor were determined by grid search with values in $[0.1, 0.05, 0.01, 0.005, 0.001, 0.0005]$. The temperature parameter τ was set to 0.07, which is a value commonly used in many previous studies. The number of negative examples, K, was fixed to a value of 5000; preliminary experiments with smaller values (< 5000) showed performance degradation, while we did not experiment with larger values due to computational issues. The best model for each configuration was selected based on the validation loss. An important fact to note is that during WaPIRL pre-training, a memory bank is only maintained for the training data, thus creating a discrepancy between training and validation loss calculations. During model validation, we modestly used the memory values of the training set to sample negative examples. Also, the memory representation m_i, which does not exist for validation samples, was substituted with its original representations z_i, while calculating (6).

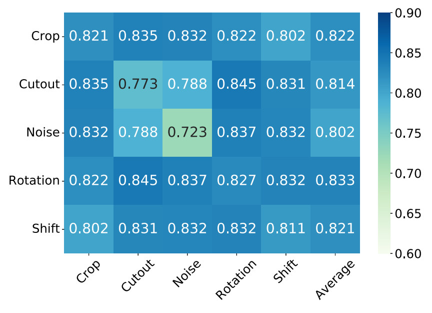
As with supervised fine-tuning, we split the labeled dataset into three subsets, while preserving the class label distribution: 80% for training, 10% for validation, and 10% for testing. To investigate how our method scales with the size of labeled data, we followed the standard evaluation protocol used in semi-supervised learning research and varied the size of labeled training data from a minimum of 1383 (1%) to a maximum of 138360 (100%) samples. For equal comparisons, we performed fine-tuning for a total of 100 epochs with a batch size of 256, regardless of the size of the training set. We used SGD to optimize the parameters with a momentum factor of 0.9, a base learning rate of 0.01, and a weight decay factor of 0.001. These values were also determined by grid search. Similar to self-supervised pre-training, the learning rate was decayed by following a half cosine learning rate schedule. We selected the model yielding the lowest validation error as the best performing model and reported evaluation



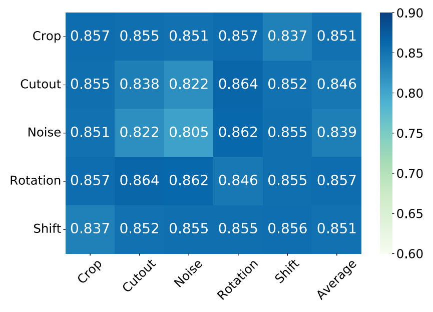
(a) 1% labels



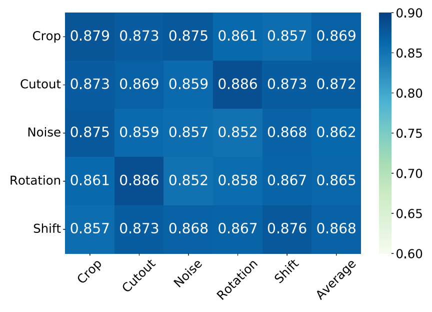
(b) 5% labels



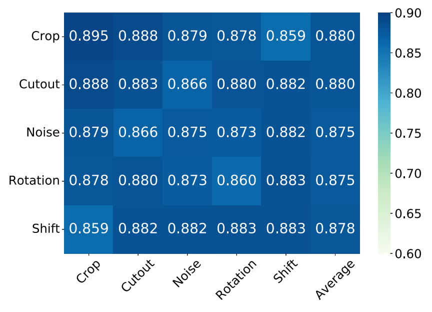
(c) 10% labels



(d) 25% labels



(e) 50% labels



(f) 100% labels

Fig. 3. Performance comparisons of WaPIRL with ResNet-18, trained with different compositions of data augmentations. (a) 1%, (b) 5%, (c) 10%, (d) 25%, (e) 50%, and (f) 100% of labeled training data is used. In each subfigure, the rightmost column contains the row-wise averages.

TABLE III
COMPARISONS OF DIFFERENT CLASSIFICATION MODELS, IN TERMS OF THE MACRO F_1 SCORE. THE BEST VALUES FOR EACH PROPORTION OF LABELED DATA IS IN **BOLD**. WE REPORT THE MEDIAN VALUES OVER TEN REPEATED TRIALS WITH DIFFERENT RANDOM SEEDS

Model	Labeled Data					
	1,383 (1%)	6,918 (5%)	13,836 (10%)	34,590 (25%)	69,180 (50%)	138,630 (100%)
WMFPR	0.498	0.692	0.717	0.726	0.749	0.767
WMDPI + RF	0.489	0.674	0.721	0.784	0.805	0.829
WMDPI + GB	0.412	0.606	0.623	0.667	0.704	0.727
WMDPI + SVE	0.572	0.696	0.717	0.757	0.785	0.801
CNN-WDI	0.630	0.790	0.839	0.851	0.858	0.877
d-CAE (ResNet-18)	0.403	0.706	0.745	0.830	0.850	0.869
d-CAE (ResNet-50)	0.411	0.856	0.788	0.845	0.861	0.859
WaPIRL+Crop (AlexNet)	0.708	0.783	0.831	0.849	0.870	0.885
WaPIRL+Crop (VGG16)	**0.741**	**0.815**	**0.839**	**0.864**	**0.880**	**0.897**
WaPIRL+Crop (ResNet-18)	0.693	0.792	0.821	0.857	0.879	0.895
WaPIRL+Crop (ResNet-50)	0.684	0.794	0.823	0.857	0.873	0.892

metrics calculated on the test set. Fully supervised baselines follow the same hyperparameter settings. All experiments were implemented with PyTorch 1.6.0 and conducted on a single NVIDIA Tesla V100 GPU. The implementations are available at https://github.com/hgkahng/WaPIRL.

C. Downstream Classification Performance

First of all, we demonstrated the superiority of WaPIRL by comparing its performance with fully supervised baselines. We reported median classification accuracies of ten repeated trials in terms of the macro F_1 score, which is the arithmetic mean of class-wise F_1 scores. Table II summarizes the results. Each column is distinguished by the size of the labeled data used during supervised training (either fully supervised or fine-tuned), and each row is distinguished by

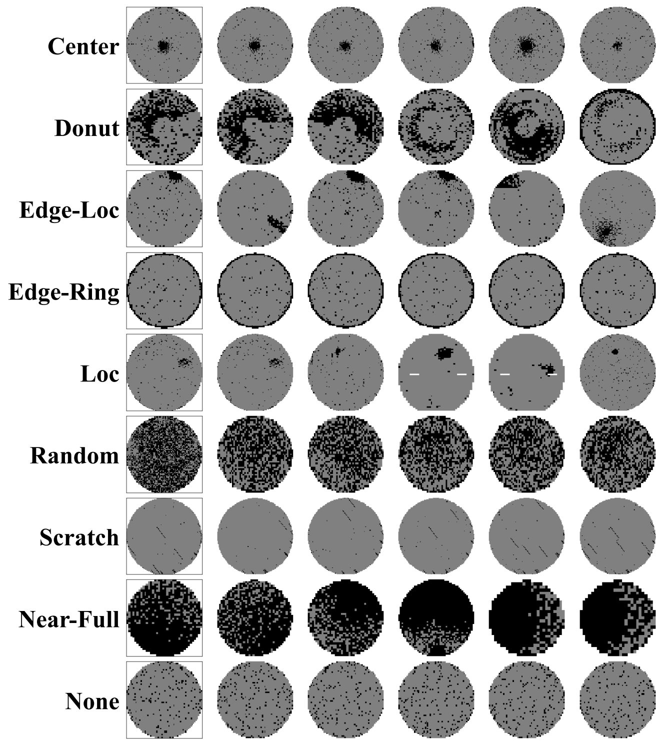
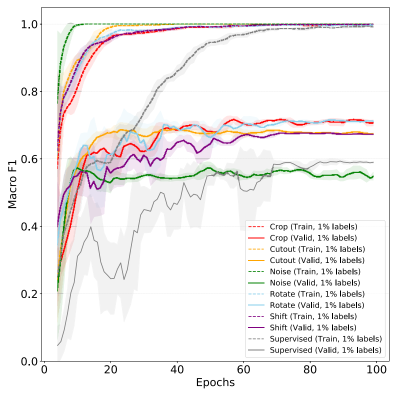
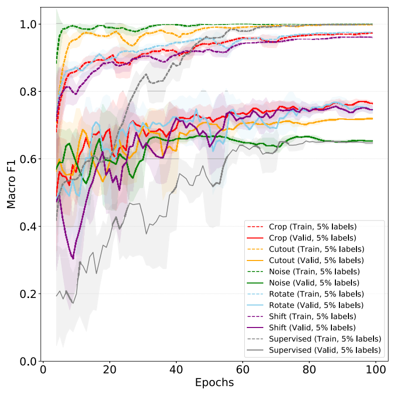


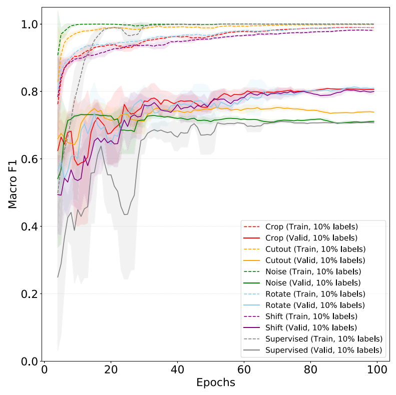
Fig. 4. Representative WBM clusters for each defect pattern class (row) obtained by five nearest neighbors matching (column) performed on the penultimate layer's output features, $z = f_\theta(x)$, of ResNet-18 (*crop*). The query WBMs are shown in the leftmost column. Queries were randomly chosen from the labeled test set. Black pixels in WBM images indicate defective dies.



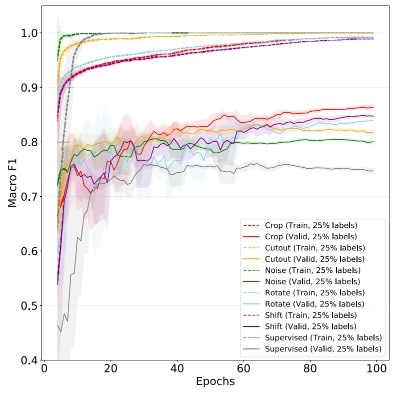
(a) 1% labels



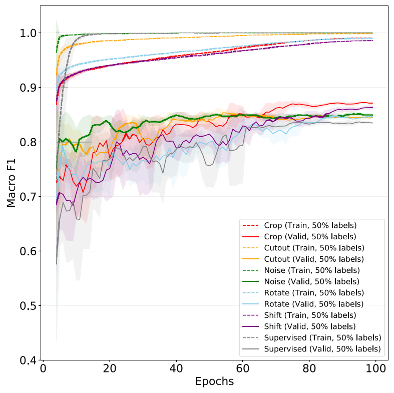
(b) 5% labels



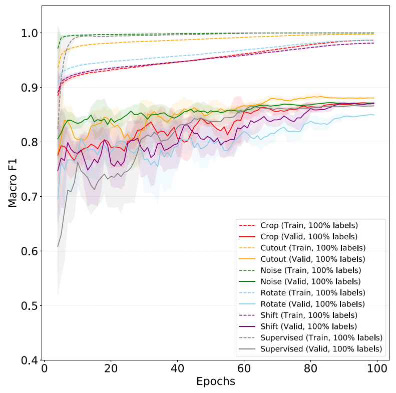
(c) 10% labels



(d) 25% labels



(e) 50% labels



(f) 100% labels

Fig. 5. Training (dashed) and validation (solid) learning curves of supervised ResNet-18 (gray) versus WaPIRL with ResNet-18 (augmentations distinguished by colors). Macro F_1 scores are plotted as a function of training epochs using different proportions of labeled data. (a) 1%, (b) 5%, (c) 10%, (d) 25%, (e) 50%, and (f) 100% of labeled training data is used. The shaded regions indicate standard deviations calculated with a rolling window size of five. Note that the y-axis limits of the first and second rows are set differently.

TABLE IV
WaPIRL Architecture Using AlexNet for the CNN Encoder. Unlike the Original Version Proposed in [5], We Adopt the Batch Normalized Variant; Local Response Normalization Layers That Follow *conv1* and *conv2* Are Removed, and a Batch Normalization Layer Is Added After Every Convolutional Layer. ReLU Nonlinearity Follows Batch Normalization, But We Omit Both From the Table for Simplicity

Module	Layer Name	Spatial Resolution	Channels	Kernel Size	Stride	Padding
WBM Input		96×96	1	-	-	-
	conv1	45×45	96	11	4	2
	maxpool1	22×22	96	3	2	0
	conv2	22×22	256	5	1	2
	maxpool2	10×10	256	3	2	0
$f_\theta(\cdot)$	conv3	10×10	384	3	1	1
	conv4	10×10	384	3	1	1
	conv5	10×10	256	3	1	1
	maxpool5	4×4	256	3	2	0
	global average pooling	1×1	256	-	-	-
$g_\phi(\cdot)$	linear + l_2-normalize	128		-	-	-
$h_w(\cdot)$	dropout(0.5) + linear + softmax	9		-	-	-

the model configuration. We observed that pre-training with WaPIRL outperformed supervised baselines for all proportions of labeled data, regardless of the choice of the encoder architecture. The improvements brought by self-supervised pre-training are significant, especially in a low labeled data regime. For instance, the VGG16 version of WaPIRL trained

TABLE V
WaPIRL Architecture Using VGG16 for the CNN Encoder. The Number Multiplied Next to Convolutional Layer Names (i.e., *conv1*($\times$2)) Indicate the Number of Repeats. Unlike the Original Version Proposed in [8], We Adopt the Batch Normalized Variant; a Batch Normalization Layer is Inserted Between Each Convolutional Layer and ReLU Nonlinearity. For Notational Simplicity, We Omit Batch Normalization and ReLU in the Table

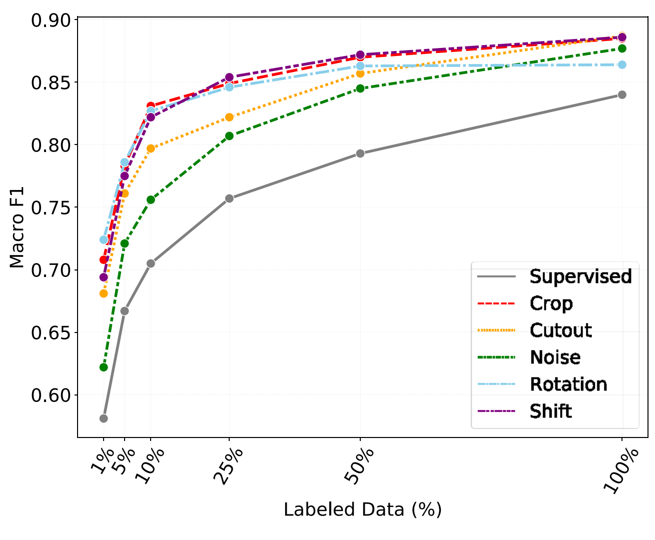
Module	Layer Name	Spatial Resolution	Channels	Kernel Size	Stride	Padding
WBM Input		96×96	1	-	-	-
	conv1 ($\times$2)	96×96	64	3	1	1
$f_\theta(\cdot)$	maxpool1	48×48	64	2	2	0
	conv2 ($\times$2)	48×48	128	3	1	1
	maxpool2	24×24	128	2	2	0
	conv3 ($\times$3)	24×24	256	3	1	1
	maxpool3	12×12	256	2	2	0
	conv4 ($\times$3)	12×12	512	3	1	1
	maxpool4	6×6	512	2	2	0
	conv5 ($\times$3)	6×6	512	3	1	1
	maxpool5	3×3	512	2	2	0
	global average pooling	1×1	512	-	-	-
$g_\phi(\cdot)$	linear + l_2-normalize	128		-	-	-
$h_w(\cdot)$	dropout(0.5) + linear + softmax	9		-	-	-

TABLE VI
WaPIRL Architecture Using ResNet-18 for the CNN Encoder. Each Convolutional Layer is Followed by Batch Normalization and ReLU Nonlinearity, But We Omit Them for Simplicity. Layers Prefixed With *'res'* Refer to Basic Residual Blocks. Values for Stride and Padding (Indicated by *) Follow the Original Setting as in [17]

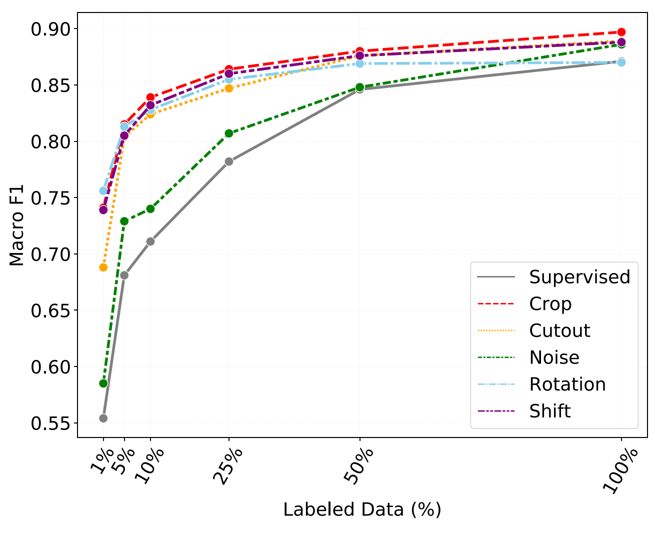
Module	Layer Name	Spatial Resolution	Channels	Kernel Size	Stride	Padding
WBM Input		96×96	1	-	-	-
	conv1	48×48	64	7	2	3
$f_\theta(\cdot)$	maxpool	24×24	64	3	2	1
	res2	24×24	64	3	*	*
	res3	12×12	128	3	*	*
	res4	6×6	256	3	*	*
	res5	3×3	512	3	*	*
	global average pooling	1×1	512	-	-	-
$g_\phi(\cdot)$	linear + l_2-normalize	128		-	-	-
$h_w(\cdot)$	dropout(0.5) + linear + softmax	9		-	-	-

with *rotation* augmentation yielded a 20.1% absolute increase in accuracy when fine-tuned on only 1% of labeled samples. Although the effects of self-supervised pre-training tend to diminish as the size of labeled data increases, we were still able to witness a substantial performance gap ($2 \sim 5\%$ with 100% labels). The decline is obvious to some extent, in that fully supervised baselines already have a sufficient amount of labeled WBMs to learn from [38]. Nonetheless, we believe that further research directions should head towards widening the performance gap in the presence of more labels.

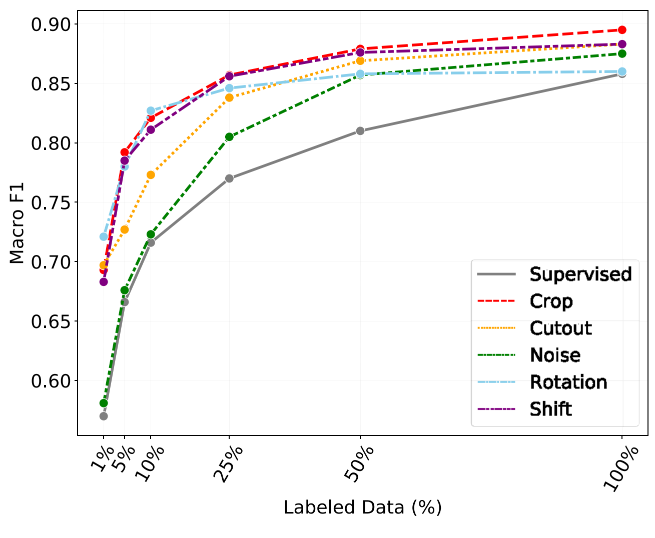
Moreover, we observed that downstream classification performance differs considerably depending on the choice of data augmentation used during WaPIRL pre-training. We visualize the numbers of Table II in Fig. 6 to help better understanding of the aspects. Overall, we claim that *cropping* is the most suitable among the five augmentation methods applied individually; it consistently showed leading performance over both different architectures and proportions of labeled data. Although *rotation* is also comparable in a low data regime, the performance tends to saturate beyond a certain level of label information. Our interpretation is that the data diversity induced by *rotation* is no longer beneficial, when a sufficient amount of labeled WBMs with various orientations are already provided. To further study the importance of data augmentation, we investigated the performance of our method when pre-trained with compositions of two different



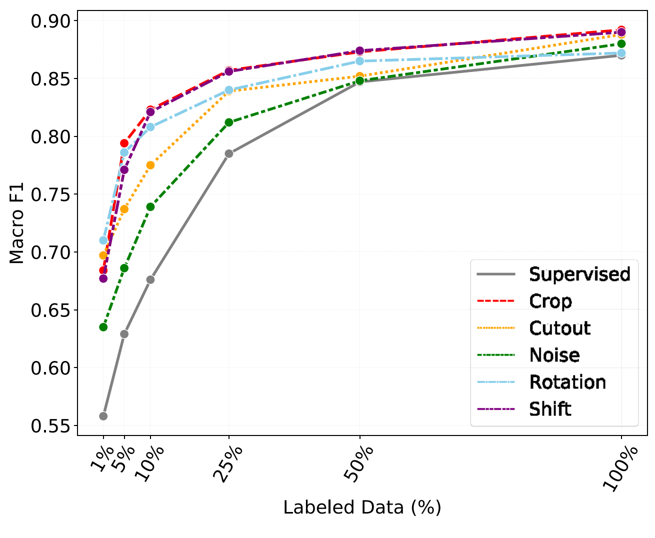
(a) AlexNet



(b) VGG16



(c) ResNet-18



(d) ResNet-50

Fig. 6. Graphical version of Table II. (a) AlexNet, (b) VGG16, (c) ResNet-18, and (d) ResNet-50. Data augmentations are distinguished by colors.

data augmentations. For illustration, we pre-trained WaPIRL with the ResNet-18 model under equal settings. As shown in Fig. 3, it is noticeable that improvements can be made with multiple augmentations when mixed properly. Conforming to the findings from single augmentation experiments, we observed *cropping* to play a the pivotal role in improving overall classification performance. The effects of mixing *rotation* with other modes is also noticeable with limited labels. On the other hand, almost no improvements were witnessed when combining *shifting* with other modes. In particular, it is evident that *cropping+shifting* deteriorates classification performance for all labeled data settings. We suspect that this combination causes too much distortion resulting in the loss of discriminative information.

In Table III, we provide comparisons between WaPIRL and other recently proposed methods that were devised for WBM classification and were tested on the same WM-811K dataset: WMFPR [10], WMDPI [33], and CNN-WDI [39]. The methods were reproduced based on their respective papers. We also conducted experiments for denoising convolutional autoencoders (d-CAE), which is a popular approach for pre-training CNNs. After pre-training a d-CAE model, a linear softmax classifier was substituted for the decoder and fine-tuned on labeled data. We followed standard practices for the design of the decoder architecture; the order of layers was symmetric to the encoder, and convolutional layers were replaced with transpose convolutional layers. It is clearly observable that our proposed method, WaPIRL, outperforms all other baseline models.

D. Analysis

To assess the quality of feature representations learned by WaPIRL, we performed nearest neighbor matching with feature vectors obtained from the penultimate layer f_θ of ResNet-18 trained with *crop* augmentation. No fine-tuning with labeled data is performed. We used the normalized Euclidean distance to measure pairwise similarities in the

TABLE VII
WaPIRL Architecture Using ResNet-50 for the CNN Encoder. Each Convolutional Layer is Followed by Batch Normalization and ReLU Nonlinearity, But We Omit Them for Simplicity. Layers Prefixed With *'res'* Refer to Bottleneck Residual Blocks. Values for Kernel Size, Stride, and Padding (indicated by *) Follow the Original Setting as in [17]

Module	Layer Name	Spatial Resolution	Channels	Kernel Size	Stride	Padding
WBM Input		96×96	1	-	-	-
	conv1	48×48	64	7	2	3
$f_\theta(\cdot)$	maxpool	24×24	64	3	2	1
	res2	24×24	256	*	*	*
	res3	12×12	512	*	*	*
	res4	6×6	1024	*	*	*
	res5	3×3	2048	*	*	*
	global average pooling	1×1	2048	-	-	-
$g_\phi(\cdot)$	linear + l_2-normalize	128		-	-	-
$h_w(\cdot)$	dropout(0.5) + linear + softmax	9		-	-	-

TABLE VIII
Classification Performances Using AlexNet, VGG16, ResNet-18, and ResNet-50 Encoders, Reported in Terms of the Area Under Precision-Recall Curve (AUPRC) Scores. For Each Encoder Architecture and Labeled Data Setting, the AUPRC Score for the Best Performing Augmentation Method is in **bold**. For Each Proportion of Labeled Data, Scores of the Best Performing Model Across Architectures Are underlined (One for Each Column)

Encoder Architecture	Training Method	Augmentation	Labeled Data					
			1,383 (1%)	6,918 (5%)	13,836 (10%)	34,590 (25%)	69,180 (50%)	138,630 (100%)
AlexNet	Supervised Baseline		0.425	0.502	0.522	0.527	0.548	0.617
	WaPIRL	Crop	0.517	0.551	0.582	0.609	0.581	0.574
		Cutout	0.483	0.581	0.611	0.627	**0.662**	**0.649**
		Noise	0.448	0.519	0.562	0.610	0.597	0.611
		Rotation	0.538	0.573	**0.616**	0.615	0.582	0.573
		Shift	**0.539**	**0.595**	0.584	**0.663**	0.625	0.608
VGG16	Supervised Baseline		0.442	0.503	0.521	0.584	0.585	0.588
	WaPIRL	Crop	**0.607**	0.628	0.662	0.671	0.615	0.617
		Cutout	0.494	0.607	0.671	0.598	0.627	0.663
		Noise	0.455	0.605	0.583	0.614	0.568	0.618
		Rotation	0.565	0.581	0.618	0.640	0.587	0.607
		Shift	0.559	**0.644**	**0.675**	**0.706**	**0.719**	**0.694**
ResNet-18	Supervised Baseline		0.436	0.536	0.563	0.559	0.588	0.659
	WaPIRL	Crop	**0.548**	**0.630**	0.652	0.677	0.581	0.665
		Cutout	0.537	0.605	0.616	0.638	0.648	0.617
		Noise	0.483	0.579	0.614	0.660	0.628	0.633
		Rotation	0.545	0.572	0.623	0.637	0.611	0.596
		Shift	0.540	0.598	**0.653**	**0.690**	**0.709**	**0.755**
ResNet-50	Supervised Baseline		0.454	0.530	0.518	0.598	0.677	0.650
	WaPIRL	Crop	0.535	0.609	0.607	0.674	0.615	0.672
		Cutout	0.507	0.583	0.627	0.591	0.599	0.619
		Noise	0.503	0.564	0.595	0.659	0.670	0.609
		Rotation	**0.547**	0.592	0.645	0.652	0.632	0.608
		Shift	0.505	**0.620**	**0.649**	**0.708**	**0.738**	**0.703**

extracted feature space. The five nearest neighbors obtained for each WBM defect pattern class are visualized in Fig. 4. The results are straightforward in that visually similar WBMs are collected for all defect pattern classes. Notably, WBMs with semantically similar defect patterns located in different regions are acquired for the 'Donut', 'Edge-Loc', 'Loc', and

'Scratch' classes. We claim that our proposed method is capable of effectively learning a rich set of features that well characterizes the data manifold without any labeled data.

Moreover, to confirm that WaPIRL provides useful initial parameters in terms of downstream classification, we compared the training and validation learning curves of fine-tuned ResNet-18 models and its fully supervised counterpart in Fig. 5. It can be observed that for all proportions of labeled data, our method already achieves a remarkably higher accuracy after only a few epochs of training. In addition, pre-trained models exhibited faster convergence, along with training stability in terms of variance. This in turn proves our original conjecture that WaPIRL pre-training provides useful initial parameters.

V. Conclusion

In this study, we investigated the application of self-supervised pre-training to the WBM domain and proposed the WaPIRL framework to address the task of WBM defect pattern classification. By leveraging a vast amount of unlabeled WBM data readily available yet neglected, the proposed method contributes to substantial improvements in downstream classification performance. In our experiments, we observed that the performance gains resulting from self-supervised pre-training are especially significant in low data regimes. By considering five different data augmentations and their compositions, we demonstrated that the choice of data augmentation is a critical factor governing model performance.

As we expect our method to benefit the semiconductor manufacturing industry in the areas of yield enhancement and failure inspection, we wish to note that the data augmentations must be chosen carefully regarding the downstream application. In certain applications such as defect root cause analysis, some data augmentations may be inappropriate in that it could unintentionally remove valuable information by impairing the orientation of the WBMs. Unfortunately, the current work does not consider such cases as we focus on WBM classification solely.

Our work can be extended in several intriguing directions. First of all, developing a better composition of data augmentation techniques that reflect the properties of WBMs would facilitate the learning of better representations. In particular, Bayesian optimization would be a natural choice to determine the optimal set of augmentations. In addition, semi-supervised learning approaches where models are simultaneously trained on both labeled and unlabeled WBMs can be considered to prevent undesirable false negative pairs belonging to the same class. Lastly, blending the concept of novelty detection with self-supervised learning to identify unseen defect patterns would provide practical benefits for today's nano-level fabrication processes where new defect patterns are prone to emerge.

Acknowledgment

The authors would like to thank the Editor and reviewers for their useful comments and suggestions, which were greatly helpful in improving the quality of the paper.

Appendix

See Tables IV–VIII and Fig. 6.

References

[1] C.-H. Wang, W. Kuo, and H. Bensmail, "Detection and classification of defect patterns on semiconductor wafers," *IIE Trans.*, vol. 38, no. 12, pp. 1059–1068, 2006.

[2] Y.-S. Jeong, S.-J. Kim, and M. K. Jeong, "Automatic identification of defect patterns in semiconductor wafer maps using spatial correlogram and dynamic time warping," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 4, pp. 625–637 Nov. 2008.

[3] T.-S. Li and C.-L. Huang, "Defect spatial pattern recognition using a hybrid SOM–SVM approach in semiconductor manufacturing," *Exp. Syst. Appl.*, vol. 36, no. 1, pp. 374–385. 2009.

[4] M. Gutmann and A. Hyvärinen, "Noisecontrastive estimation: A new estimation principle for unnormalized statistical models," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 297–304.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[6] C.-S. Liao *et al.*, "Similarity searching for defective wafer bin maps in semiconductor manufacturing," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 953–960, Jul. 2014.

[7] C.-W. Liu and C.-F. Chien, "An intelligent system for wafer bin map defect diagnosis: An empirical study for semiconductor manufacturing," *Eng. Appl. Artif. Intell.*, vol. 26, nos. 5–6, pp. 1479–1486, 2013.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: arXiv:1409.1556.

[9] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[10] M.-J. Wu, J.-S. R. Jang, and J.-L. Chen, "Wafer map failure pattern recognition and similarity ranking for large-scale data sets," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 1, pp. 1–12 Feb. 2015.

[11] F. Adly *et al.*, "Randomized general regression network for identification of defect patterns in semiconductor wafer maps," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 2, pp. 145–152, May 2015.

[12] C. Doersch, A. Gupta, and A. A Efros, "Unsupervised visual representation learning by context prediction," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1422–1430.

[13] A. Dosovitskiy *et al.*, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1734–1747, Sep. 2016.

[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015. [Online]. Available: arXiv:1503.02531.

[15] B. Kim *et al.*, "A regularized singular value decomposition-based approach for failure pattern classification on fail bit map in a DRAM wafer," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 1, pp. 41–49 Feb. 2015.

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

[17] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[18] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016. [Online]. Available: arXiv:1608.03983.

[19] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 69–84.

[20] D. Pathak *et al.*, "Context encoders: Feature learning by inpainting," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2536–2544.

[21] G. Larsson, M. Maire, and G. Shakhnarovich, "Colorization as a proxy task for visual understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6874–6883.

[22] T. Xiao *et al.*, "Joint detection and identification feature learning for person search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 3415–3424.

[23] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," 2018. [Online]. Available: arXiv:1803.07728.

[24] K. Kyeong and H. Kim, "Classification of mixed-type defect patterns in wafer bin maps using convolutional neural networks," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 3, pp. 395–402, Aug. 2018.

[25] T. Nakazawa and D. V Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 2, pp. 309–314, May 2018.

[26] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018. [Online]. Available: arXiv:1807.03748.

[27] Z. Wu *et al.*, "Unsupervised feature learning via non-parametric instance discrimination," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3733–3742.

[28] P. Bachman, R. D. Hjelm, and W. Buchwalter, "Learning representations by maximizing mutual information across views," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 15509–15519.

[29] K. He *et al.*, "Momentum contrast for unsupervised visual representation learning," 2019. [Online]. Available: arXiv:1911.05722.

[30] J. Kim *et al.*, "Bin2Vec: A better wafer bin map coloring scheme for comprehensible visualization and effective bad wafer classification," *Appl. Sci.*, vol. 9, no. 3, p. 597, 2019.

[31] A. Kolesnikov, X. Zhai, and L. Beyer, "Revisiting self-supervised visual representation learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1920–1929.

[32] I. Misra and L. van der Maaten, "Selfsupervised learning of pretext-invariant representations," 2019. [Online]. Available: arXiv:1912.01991.

[33] M. Saqlain, B. Jargalsaikhan, and J. Y. Lee, "A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 2, pp. 171–182, May 2019.

[34] R. Wang and N. Chen, "Wafer map defect pattern recognition using rotation-invariant features," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 4, pp. 596–604, Nov. 2019.

[35] T. Chen *et al.*, "A simple framework for contrastive learning of visual representations," 2020. [Online]. Available: arXiv:2002.05709.

[36] X. Chen *et al.*, "Improved baselines with momentum contrastive learning," 2020. [Online]. Available: arXiv:2003.04297.

[37] Y. Kong and D. Ni, "A semi-supervised and incremental modeling framework for wafer map classification," *IEEE Trans. Semicond. Manuf.*, vol. 33, no. 1, pp. 62–71, Feb. 2020.

[38] A. Newell and J. Deng, "How useful is self- supervised pretraining for visual tasks?" 2020. [Online]. Available: arXiv:2003.14323.

[39] M. Saqlain, Q. Abbas, and J. Y. Lee, "A deep convolutional neural network for wafer defect identification on an imbalanced dataset in semiconductor manufacturing processes," *IEEE Trans. Semicond. Manuf.*, vol. 33, no. 3, pp. 436–444, Aug. 2020.