When Wafer Failure Pattern Classification Meets Few-shot Learning and Self-Supervised Learning

Hao Geng[1], Fan Yang[2], Xuan Zeng[2], Bei Yu[1]
[1]The Chinese University of Hong Kong [2]Fudan University
{hgeng,byu}@cse.cuhk.edu.hk, {yangfan,xzeng}@fudan.edu.cn

***Abstract*—Due to advances in semiconductor processing technologies, wafer failure pattern detection plays a key role in preventing yield loss excursion events for semiconductor manufacturing. In the recent semiconductor industry, visible surface defects are still mainly being inspected manually, which may result in inevitably erroneous classification. Many machine learning techniques-based pioneered arts in academia have been proposed to aid wafer failure pattern classification. However, few of these attach importance to unlabeled information and alleviate the data imbalanced issue. Based on these concerns, this paper designs an end-to-end wafer defect classifier that unites the few-shot learning and self-supervised learning algorithms. The aim of applying the few-shot learning paradigm is to learn representations that generalize well to the minority defect pattern classes where only a few wafer images are available, while the self-supervision information containing the intrinsic correlations of unlabeled wafer maps and their augmentations is expected to enhance the few-shot learner. The experimental results demonstrate the proposed framework has superior performance compared to cutting-edge wafer defect classification methods.**

I. Introduction

Continuous shrinkage of process technology nodes and the increasingly complicated nature of integrated circuit (IC) designs make IC manufacturing difficult, and the number of situations where yield improvement exhibits vitally is soaring. On the other hand, as the pressure for meeting high specifications increases, the probabilities of manufacturing process-based defects appearing on the surface of the wafers also increase. Wafer defects become the primary obstacles affecting product yield in IC manufacturing. Hence, wafer map defect classification and analysis, which locates defects at early fabrication stages, has become essential. Specifically, defective grains on a wafer map tend to converge into a certain distribution pattern that includes critical information. Semiconductor engineers need to identify these failure patterns in production processes to improve yield. However, in actual manufacturing, wafer map failure pattern classification and analysis are mainly carried out manually, and the related researches remain in the theoretical stage.

In academia, quite a few efforts on automating wafer map defect pattern classification have been made. The taxonomy of these approaches has two branches: 1) model-based pattern recognition, 2) feature extraction-based pattern recognition. Concerning model-based pattern recognition, a pre-defined probability distribution function is used for each pattern and the best statistical model is determined by comparing models using information criterion. Regarding the feature extraction-based defect pattern recognition, a large amount of pioneered arts [1]–[7] have been proposed, where feature design or extraction has a significant impact on the accuracy of failure pattern classification. These feature extraction-based works can be further divided into two categories: manually-crafted feature-driven and automatic feature extraction-based. For example, Wu *et al*. [1] develop radon-based and geometry-based features, while Yu *et al*. [2] define geometrical, gray, texture, and projection features, and merge the features by their proposed dimension reduction and feature extraction methods. The domain knowledge of expert engineers is in demand when manually designing the feature. Notice that some of the manually-crafted feature-based methods work in an unsupervised fashion. For instance, in [3], clusters of wafer maps are first constructed, and then the failure pattern labeling work is assigned to experienced engineers. Neither feature design nor failure labeling for wafers is fully automatic with such approaches. What's worse, the feature design part separates from the follow-up classifier, which leads the whole failure classification framework to converge to the sub-optimality.

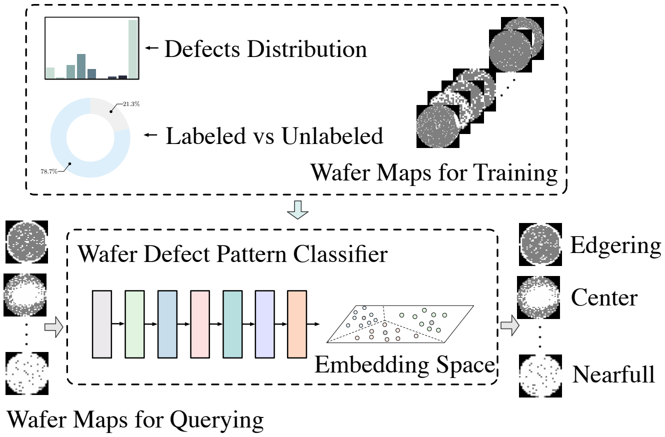


Fig. 1 The proposed wafer failure pattern classifier.

Recently, deep learning methods which get rid of manually-crafted feature design have achieved notable success in many areas, particularly in computer vision. Several works [4]–[7] have introduced deep learning techniques and verified the potential of deep convolutional neural networks (CNNs) in the field of wafer failure pattern classification. For example, in [4], an 8-layer CNN model is constructed, which achieves high recognition accuracy after calibration on synthetic wafer dataset. Yu *et al*. [5] propose a defect pattern classification model and a classification model based on CNN. Alawieh *et al*. [7] present a wafer map defect pattern classification framework using deep selective learning, featuring an integrated reject option which can further improve the model accuracy by abstaining from providing predictions for samples with high misclassification risk. Yet these arts exploit the deep learning schemes at a coarse level and seldom utilize the unlabeled information which is in the majority of available wafer map data in practice.

It could be plainly viewed that two main issues exist in previous works and the broadly used industrial benchmark for wafer map defect classification. One is that the real industry data (e.g. `WM-811K` dataset [8]) is haunted with the imbalanced distribution issue (some minority failure pattern types are dominated by other majority ones), posing great challenges for wafer defect classification models. In other words, defect classes have different frequencies of occurrence and many categories lack sufficient associated wafter maps. It results in a biased classifier where the decision boundary can be drastically altered by the majority classes. Finally, an inaccurate defect classification result would be reported by the poorly calibrated model. Although some works like [7] exploit some deep learning-based technique to generate synthetic wafer patterns, they separate the generation process from the classifier calibration, and even worse, synthesis may lead to the label perturbation. The other problem is many unlabeled wafer maps are idle in most previous arts. For the former issue, recently investigated few-shot learning techniques which are related to imbalanced learning [9] can alleviate. The goal of few-shot learning is to learn representations that generalize well to the minority defect pattern classes where only a few wafer images are available. In the setting of few-shot learning, a small amount of data from different classes are sampled evenly in one training batch, which is equivalent to the resampling tactics to combat imbalanced training data. By leveraging the self-supervised learning, the unlabeled wafer maps can come in handy. Besides, it has been revealed that self-supervised learning can be used to improve other supervised counterpart visual tasks [10]. To some extent, the wafer image itself already contains structural information that can be utilized. Therefore, the self-supervised learning technique which explores the correlations between each wafer map and augmentations of itself would have a promise. Note that the augmented versions can be acquired by applying rotation and flipping on the original wafer maps.

Based on the above discoveries, in this work, we combine the few-shot learning and self-supervised learning algorithms, and design an end-to-end wafer failure pattern classifier (visualized in Fig. 1). Apart from achieving superior performance compared to previous approaches, our proposed flow are tailored to address aforementioned challenges accompanying the task. Our main contributions are summarized as follows:

- An end-to-end wafer map defect pattern classification flow is proposed.
- The self-supervised learning technique is embedded into the developed flow to make full use of the tremendous unlabeled wafer maps, so as to reduce the human labour efforts.
- The few-shot learning paradigm is incorporated to overcome the data imbalanced issue.
- The experimental results comprehensively demonstrate the superiority of the proposed wafer defect pattern classification framework.

The rest of the paper is organized as follows. Section II introduces some prior knowledge, and then gives the problem formulation. Section III describes the detailed techniques in proposed failure pattern classification flow and sketches its whole view. Section IV presents the experimental results, followed by the conclusion in Section V.

II. Preliminaries

In this section, we first introduce some related works about the few-shot learning and the self-supervised learning. Then we describe the wafer map defect classification and the problem formulation.

A. The few-shot learning

The few-shot learning method [11]–[17] can be roughly classified into three categories [9]. The first class of methodologies include optimization-based meta-learners, like model-agnostic meta-learner [11], gradient unrolling [12], closed-form solvers [13], and convex learners [14]. The second category relies on metric-based classifiers such as matching networks [15] and prototypical networks [16]. The rest type of approaches [17] model the mapping between training data and classifier weights using a feed-forward network.

B. The self-supervised learning

The self-supervised learning [18]–[22] can be viewed as a branch of unsupervised learning since there is no manual label involved. This kind of approaches targets at learning useful representations of the input data without relying on human annotations. The recent pace of progress has increased dramatically and led to self-supervised deep representations that even surpass that of fully-supervised representations. Chen et al. [18] provide a contrastive self-supervised learning method (e.g., “SimCLR”) which defines “positive” and “negative” sample pairs and treats them differently in the loss function. He et.al [19] further treat the contrastive self-supervised learning as a dictionary look-up task and design momentum encoders to get tremendous “negative” samples. In another recent line of arts, [20] and [21], still use the contrastive learning paradigm but both the network architecture and parameter updates are modified to introduce asymmetry.

C. Wafer failure pattern classification

Generally, in semiconductor manufacturing, engineers use wafer defect maps to visualize failure patterns distributed by a large number of contaminated dies and identify potential process and tool issues. By nature, wafer failure patterns are spatial patterns that can be classified visually when treating the wafer maps as images. Therefore, keeping the wafer maps in their image representation can best preserve the defect information. A potential approach to classifying different spatial wafer defect patterns is through casting the problem as a multi-class image classification task and leveraging convolutional neural networks to handle it. Hence, our objective is to train a deep learning based-model capable of categorizing wafer defect pattern types. To this end, we propose a wafer map defect pattern classification framework using few-shot learning and self-contrastive learning strategies, which feature an end-to-end flow. With the aforementioned knowledge, our problem can be formulated as follows.

Problem 1 (Wafer Map Defect Pattern Classification)**.** *Given a collection of wafer maps containing distinct patterns, the objective of our wafer failure pattern classification flow is to train a model in the scenario of data imbalance and lacking human annotations, and classify all wafer failure patterns as accurately as possible.*

III. The Developed Detection Flow

Wafer failure pattern classification acts an instrumental role in boosting the yield. Our main interest is not only to identify the failure patterns for a single wafer, but also provides insights and potential solutions to alleviate the real industrial dilemma with rare labeled data and imbalanced data distribution. Towards this goal, we aim at classifying the wafer failure patterns in the absence of any prior information from wafer engineers and fab. In the proposed design, there is two data source: the labeled and unlabeled wafer map sources. The proposed learning model is comprised of a shared backbone working as a wafer map feature extractor and two learning tasks. One task is for few-shot classification which targets alleviating the wafer data imbalanced issue, and the other is self-supervised learning on unlabeled wafer images.

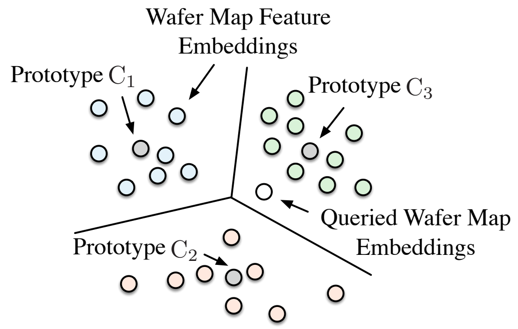


Fig. 2 The illustration of the Prototypical network-based 8-shot learner.

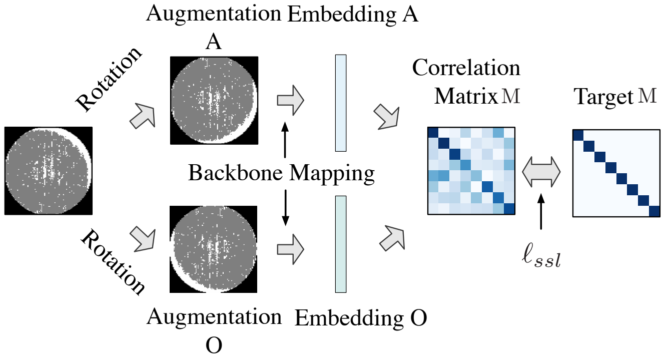


Fig. 3 The visualization of the self-supervised learning.

A. The few-shot learner

The few-shot learning learns representations that generalize well to the minority failure pattern types where few wafer images are available. This kind of property is naturally compatible with our case.

Before describing the proposed few-shot learner, we first clarify some definitions. Assume a small training set of N labeled examples $\mathcal{D}_{train} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$ consisting of pairs of wafer images $\boldsymbol{x}_i$ and $y_i \in \{1, \ldots, K\}$ the corresponding defect type. In the few-shot learning setting, a support set $\mathcal{D}_s$ and a query set $\mathcal{D}_q$ are sampled from the $\mathcal{D}_{train}$ per training batch (or called episode). Particularly, support subset $\mathcal{D}_k$ denotes the subset of $\mathcal{D}_s$ labeled with wafer defect type k. $N_C \leq K$ is the number of classes per batch with N_S the number of support wafer examples per class (N_S is usually small) and N_Q the number of query examples for each class. In one training batch, the wafer map embeddings are trained to predict the labels of $N_Q \times N_C$ query wafer maps conditioned on $N_S \times N_C$ support wafer maps using a certain classifier. As is observed, the few-shot learner is built upon the balanced data, which is analogous to the resampling strategy to resolve imbalanced training data issue.

Generally, we exploit the Prototypical few-shot learning flow which comprises the backbone $f(\cdot;\boldsymbol{\phi})$ parameterized by learnable $\boldsymbol{\phi}$ and the classifier g as our few shot learner. The overall mapping from the input wafer map space to the label space is defined as $h(\cdot;\theta) = g\ \circ f(\boldsymbol{x};\boldsymbol{\phi})$, where $\circ$ stands for functional composition. The Prototypical network computes a vector representation $\boldsymbol{c}_k \in \mathbb{R}^M$, termed as `prototype`, of the central of each class through the backbone network $f(\cdot;\boldsymbol{\phi})$. Specifically, after the vector-based features are extracted for the support examples, each `prototype` is computed by taking the mean of the embedded support wafer map vectors belonging to the associated defect type:

$$\boldsymbol{c}_k := \frac{1}{|\mathcal{D}_k|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_k} f(\boldsymbol{x}_i; \boldsymbol{\phi}) \tag{1}$$

Then, given a distance function $d : \mathbb{R}^M \times \mathbb{R}^M \to [0, +\infty)$, the empirical loss function $\ell(\hat{y}, y)$ in Equation (2) is minimizing the negative log-probability of the generated distribution over pattern types for a query wafer image $\boldsymbol{x}$ with corresponding label y.

$$\begin{aligned} \ell(\hat{y}, y) &:= -\log p(y = k \mid \boldsymbol{x}, \boldsymbol{\phi}) \\ &= -\log \frac{\exp(-d(f(\boldsymbol{x};\boldsymbol{\phi}), \mathbf{c}_k))}{\sum_{k'=1}^{N_C} \exp(-d(f(\boldsymbol{x};\boldsymbol{\phi}), \mathbf{c}_{k'}))}. \end{aligned} \tag{2}$$

The query wafer defect map is classified based on $p(y = k \mid \boldsymbol{x}, \boldsymbol{\phi})$ which is the softmax function over certain distances to the prototypes in the embedding space. The squared Euclidean distance is adopted as the distance metric, which is defined as:

$$d(f(\boldsymbol{x};\boldsymbol{\phi}), \mathbf{c}_k) := \|f(\boldsymbol{x};\boldsymbol{\phi}) - \mathbf{c}_k\|_2^2. \tag{3}$$

We visualize the mathematical principle of the Prototypical network in Fig. 2 where $N_S = 8$ and $N_C = 3$.

The loss of few-shot learning ℓ_{few} for one batch is minimizing the empirical loss $\ell(\hat{y}_j, y_j)$ on the whole query set along with a suitable regularization r:

$$\ell_{few} := \sum_{\substack{j=1, \\ (\boldsymbol{x}_j, y_j) \in \mathcal{D}_q}}^{N_Q \times N_C} \ell(\hat{y}_j, y_j) + r. \tag{4}$$

A commonly used regularizer is the ℓ_2 norm of the parameters of the functions.

B. The self-supervised learner

The categories of wafer defect collected via IC engineers' experience are expensive and hard to scale up. To this end, making full use of unlabeled wafer images is expected. Consequently, we embed the self-supervised learning paradigm into our wafer failure pattern classification flow.

The self-supervision method operates on joint embeddings of input wafer image and its augmentation. More specifically, it produces two augmented views for all wafer images in a batch sampled from a wafer dataset. The two kinds of representations are projected into an embedding space where self-supervised loss is applied. For a better understanding, we sketch the diagram of the self-supervised learning in Fig. 3. The architecture consists of three components. First is the data augmentation module in which any given wafer map is transformed into two correlated views of itself. For wafer map detection, we can apply the following simple augmentations: rotation, top-bottom and left-right flipping. These augmentations would not change the latent categories of failure patterns. The second part is the backbone network with its replication as its counterpart. They extract representation vectors from an original wafer map and its augmented pattern. The rest module is the self-supervised loss function denoted as ℓ_{ssl} where "ssl" stands for "self-supervised". The mathematical expression of loss function is written in Equation (5).

$$\ell_{ssl} := \sum_i (1 - \boldsymbol{M}_{ii})^2 + \lambda \sum_i \sum_{j \neq j} \boldsymbol{M}_{ij}^2, \tag{5}$$

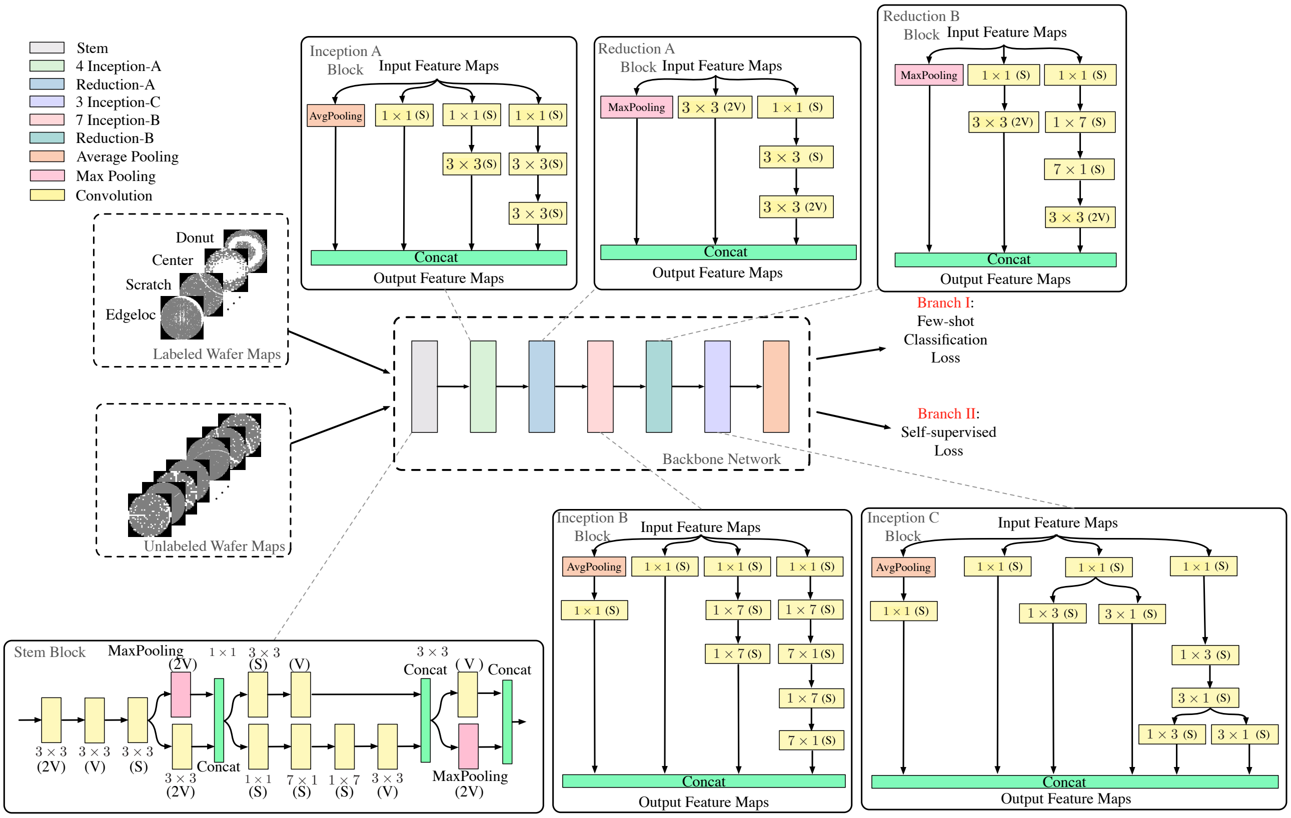


Fig. 4 The network architecture of the proposed wafer failure pattern classification framework.

where λ is a positive constant trading off the importance of the first and second terms of the loss, and $\boldsymbol{M}$ stands for the correlation matrix which computes the correlations between the outputs of the two identical networks along the batch dimension. Assuming the latent embeddings of two input views are e^o and e^a, the element $\boldsymbol{M}_{ij}$ in $\boldsymbol{M}$ is computed via Equation (6).

$$\boldsymbol{M}_{ij} := \frac{\sum_b e^o_{b,i} z^a_{b,j}}{\sqrt{\sum_b \left(e^o_{b,i}\right)^2}\sqrt{\sum_b \left(e^a_{b,j}\right)^2}} \tag{6}$$

By trying to force the diagonal elements of the correlation matrix to equal to 1 (i.e. perfect correlation), the first term of the objective function in Equation (5) makes the representation invariant to the augmentations applied. The second term, attempting to equate the off-diagonal elements of the correlation matrix to 0, de-correlates the different vector components of the representation. This decorrelation removes the redundancy between output units. As a result, the outputs contain non-redundant information about the input wafer map. Mathematically, Equation (5) can be treated as an instantiation of the information bottleneck principle [23].

In a netshell, the self-supervised learning flow first produces augmented views for all wafer maps of a batch sampled from training dataset. Then, the batches of augmented views are fed into the proposed backbone, producing batches of latent wafer map embeddings of two views respectively. Ultimately, the self-supervised learning loss is computed and back-propagated.

C. The overall architecture design

Our framework consists of a shared backbone and two branches for the few-shot learner and self-supervised learner. During training, two sources of wafer maps (i.e. labelled and unlabeled) are fed into the defect classifier for calibration. The backbone is expected to project the labeled wafer map to an embedding space in which the few-shot learner calculates softmax function for classification and maps another input from the unlabeled source to space where the self-supervised loss is computed. Thereby, the shared backbone plays a critical role in the framework.

Considering that different kinds of defect patterns on the wafer maps have different geometrical properties including size, shape, etc., a single-sized kernel is inflexible to capture such rich feature information. On the other hand, the success of applying inception mechanism into the layout hotspot detection [24] has been witnessed. The Inception-v4 [25] deep convolutional neural network (i.e. $f(\cdot;\phi)$) is adopted by us as the shared backbone. By using the Inception blocks, the backbone goes wider and deeper. It has the potential to achieve a more robust wafer map feature expression and a higher classifier accuracy comparing to a shallow one. We show the whole network architecture in Fig. 4. Besides, we supplement Fig. 4 with some explanations to improve the readability. For example, the rectangle with $3(V)$ stands for the convolutional operator with a kernel sized 3 and a “valid” padding style (input patch of each unit is fully contained in the previous layer and the grid size of the output activation map is reduced accordingly), while the one with $3(S)$ denotes that the convolutional computing is performed via a

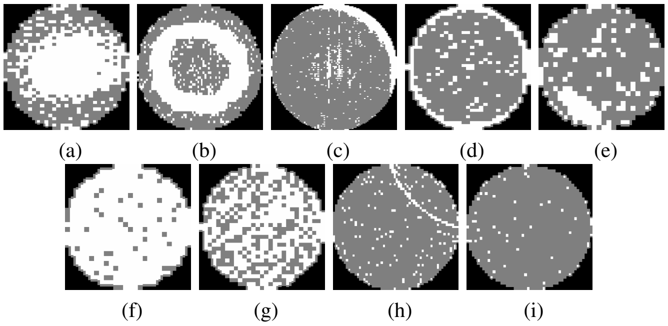


Fig. 5 The 9 kinds of wafer map pattens in the benchmark [8]: (a) Center; (b) Donut; (c) Edge-Loc; (d) Edge-Ring; (e) Location; (f) Near-Full; (g) Random; (h) Scratch; (i) None (the defect-free wafer pattern).

TABLE I Benchmark Statistics

Categories	Training Set		Testing Set	
	#	Percent (%)	#	Percent (%)
`Center`	2576	2.48	1718	2.48
`Donut`	333	0.32	222	0.32
`Edge-Loc`	3113	2.99	2076	3.00
`Edge-Ring`	5808	5.60	3872	5.60
`Location`	2155	2.08	1438	2.08
`Near-Full`	89	0.09	60	0.09
`Random`	519	0.50	347	0.50
`Scratch`	715	0.69	478	0.69
`None`	88459	85.25	58972	85.25
`Total`	103767	100	69183	100

3 kernel in a "same" padding fashion, meaning their output grid matches the size of the corresponding input. It is worth mentioning that the expression $3(2V)$ has the similar meaning with $3(V)$ except for the stride value. The former refers to a 2 stride value, whilst the latter indicates the stride value of 1.

As aforementioned, our end-to-end failure pattern classification framework has two jointly performed tasks sharing the same wafer pattern feature extractor. We combine the few-shot learner loss Equation (4) and self-supervised loss ℓ_{ssl} constituting a total loss function:

$$\mathcal{L} := \ell_{few} + \alpha \ell_{ssl}. \tag{7}$$

In our case, the trade-off coefficient α is simply set to be 1. To some extent, the self-supervised losses act as a data independent regularizer for representation learning. Via back-propagation, the gradient summation of two tasks, $\nabla_{\phi}\ell_{few}(\cdot;\phi) + \nabla_{\phi}\ell_{ssl}(\cdot;\phi)$, updates the backbone collectively. It is expected that the wafer pattern extraction by backbone is guided by a joint learning and in turn, the two branches benefit from the sharing backbone. Our flow is associated with multi-task learning which is a class of techniques that train on multiple task objectives together to improve each one. Previous arts in representation learning [26] and computer vision [27] literature have demonstrated moderate benefits by combining tasks mathematically and experimentally (i.e. $\mathcal{L}(f(\cdot;\phi)) < \sum_{t=1}^{T} \mathcal{L}_{t=i}(f_i(\cdot;\phi_i))$ with T tasks and ϕ_i the exclusive model parameters of i-th task).

IV. Experimental Results

A. *Experimental Settings and Benchmark*

The implementation of our framework is in `Python` with the `TensorFlow` library [28], and we test it on a platform with the Xeon Silver 4114 CPU processor and Nvidia TITAN Xp Graphic card. To verify the effectiveness and efficiency of our failure pattern classifier, one open industrial benchmark, `WM-811K` [8], is employed. There are 811457 wafer images from 46293 lots covering 9 categories of patterns in this industrial dataset where only 172950 images are with expert annotations. Analogous to wafer map translating schemes in the stat-of-the-art work [7], we use single-channel, grey-scale images to represent wafer maps. Towards a better understanding, we exemplify 9 wafer map visualizations containing 8 kinds of wafer failure patterns and defect-free wafer pattern in Fig. 5. As can be seen, each wafer image has 256 × 256 pixels with 3 pixel levels: 0, 127 and 255. Locations with pixel level 0 (i.e., the black pixels in wafer images) are those not part of the wafer. Grey pixels with pixel level 127 indicate die locations with a passing label, while white pixels represent those with a fail label. It is conspicuous that wafer defects of distinct categories have different defect distribution patterns. For example, the defects in the `Center` category are concentrated around or near the center of the wafer in circular or ring-like patterns, and the defects in the `Location` class are clustered on the edge of the wafer but do not behave linear or curvy characteristics, and the contaminants in the `Random` type are almost randomly distributed across the entire surface of the wafer. In general, these classes of defects have known possible causations. For instance, the `Center` class is caused due to abnormality of RF (radio frequency) power, abnormality in liquid flow, or abnormality in liquid pressure, while the `Location` type is generated by silt valve leak, abnormality during robot handoffs or abnormality in the pump. The `Random` category is likely brought about by contaminated pipes, abnormality in showerhead or abnormality in control wafers.

We re-arrange the `WM-811K` dataset as per the requests of the problem formulation and our classification flow. Note that full labeled wafer maps are used and 50000 unlabelled wafer maps are also exploited for training. We summarize the statistics of the dataset in TABLE I where unlabelled training wafer maps are not listed. The whole labeled wafer maps is split into $0.6 : 0.4$ for training and testing. Additionally, it is manifest that the number of defect data in the "Training Set" and "Testing Set" utilized in our work differs from those in prior arts. Because it is important to keep the inherent imbalanced data distribution in the training and testing dataset (e.g., dominated by the None class) so that the real industrial scenario can be mimicked.

This is natural that the classical accuracy (i.e. micro-average accuracy) is not proper for the imbalanced multi-label classification problem. In view of that a model that is competent to learn a majority class (i.e. `None` pattern) can achieve very high accuracy, while may still behave poorly on the failure pattern classes which are more important for yield analysis. We adopt the confusion matrix to fully evaluate the true states of the model performance. Besides, we also further calculated precision, recall, F_1 score to quantitatively estimate the performance. The definitions of the three metrics are written as follows. Precision is the ratio between the number of wafer defects of one defect type being correctly classified and all wafer defect patterns identified in this type. Recall (a.k.a. hit rate) is the fractional measure of the number of wafer defects of one defect type being accurately classified and all wafer failure patterns belonging to this type. The F_1 score is a trade-off between the above two metrics. When the predicted results and the actual results are close to each other, F_1 score uplifts.

TABLE II Comparison with state of the arts

Defect Pattern	TSM'15 [1]			DAC'20 [7]			Ours		
	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
Center	0.661	0.861	0.748	**0.949**	0.942	**0.945**	0.736	**0.950**	0.830
Donut	0.729	0.459	0.564	0.798	0.748	0.772	**0.806**	**0.842**	**0.824**
Edge-Loc	0.453	0.577	0.507	**0.739**	0.690	0.714	0.647	**0.802**	**0.716**
Edge-Ring	0.611	0.908	0.731	0.992	0.950	**0.970**	**0.992**	0.921	0.955
Location	0.533	0.346	0.420	0.191	0.627	0.293	**0.605**	**0.720**	**0.658**
Near-Full	0.254	**0.867**	0.392	0.697	0.383	0.495	**0.810**	**0.867**	**0.840**
Random	0.412	0.101	0.162	0.608	0.553	0.579	**0.816**	**0.652**	**0.724**
Scratch	**0.835**	0.339	0.482	0.127	0.287	0.176	0.474	**0.701**	**0.565**
None	0.973	0.940	0.956	0.985	0.927	0.955	**0.986**	**0.967**	**0.977**
Macro-average	0.607	0.600	0.551	0.676	0.679	0.656	**0.764**	**0.825**	**0.788**
Ratio	0.795	0.727	0.700	0.885	0.823	0.832	**1.000**	**1.000**	**1.000**

TABLE III The performance of the defect classifier proposed by [1] in a confusion matrix.

Defect Pattern	Center	Donut	Edge-Loc	Edge-Ring	Location	Near-Full	Random	Scratch	None
Center	1479	11	45	0	74	2	10	8	89
Donut	35	102	5	0	29	1	26	3	21
Edge-Loc	56	0	1198	187	16	44	3	7	565
Edge-Ring	0	0	128	3516	0	0	5	3	220
Location	190	4	291	6	498	0	0	11	438
Near-Full	3	0	0	0	0	**52**	5	0	0
Random	0	5	12	67	14	8	35	0	206
Scratch	147	3	23	0	27	98	1	162	17
None	327	15	945	1977	277	0	0	0	55431

TABLE IV The performance of the defect classifier proposed by [7] in a confusion matrix.

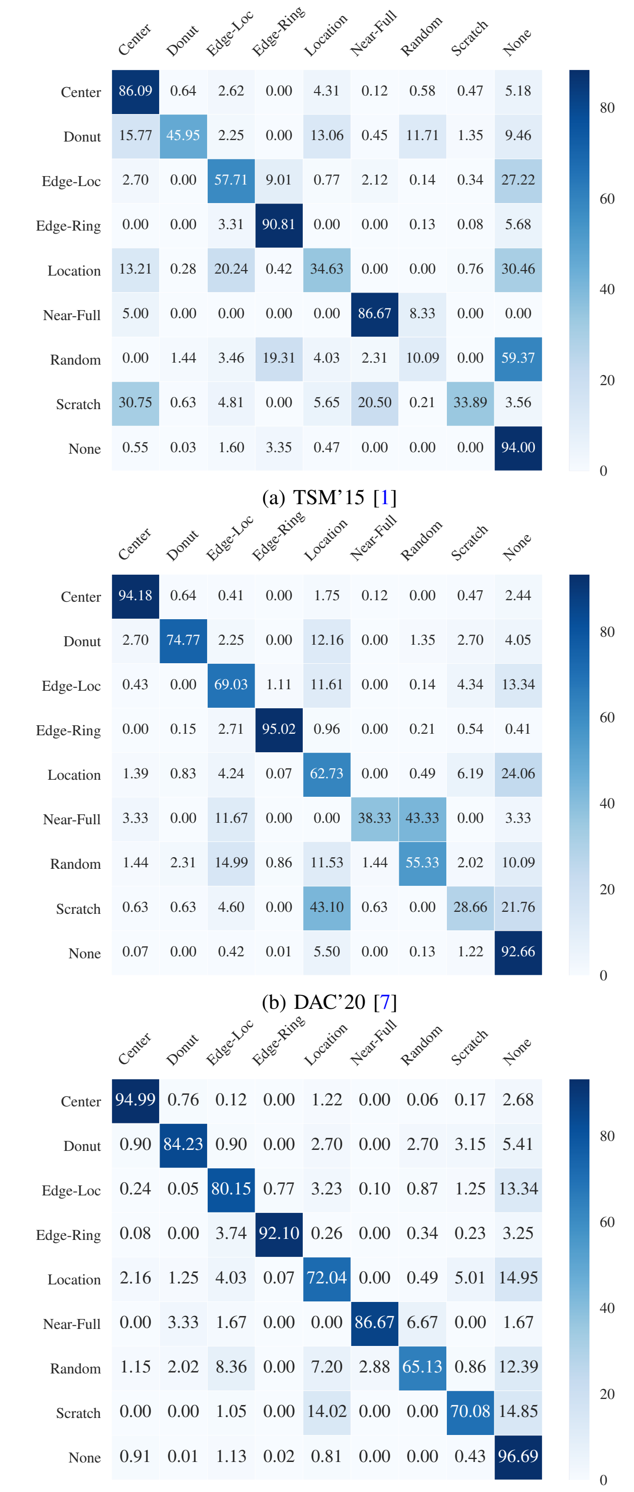
Defect Pattern	Center	Donut	Edge-Loc	Edge-Ring	Location	Near-Full	Random	Scratch	None
Center	1618	11	7	0	30	2	0	8	42
Donut	6	166	5	0	27	0	3	6	9
Edge-Loc	9	0	1433	23	241	0	3	90	277
Edge-Ring	0	6	105	**3679**	37	0	8	21	16
Location	20	12	61	1	902	0	7	89	346
Near-Full	2	0	7	0	0	23	26	0	2
Random	5	8	52	3	40	5	192	7	35
Scratch	3	3	22	0	206	3	0	137	104
None	42	2	246	4	3241	0	77	719	54641

TABLE V The performance of our classifier in a confusion matrix.

Defect Pattern	Center	Donut	Edge-Loc	Edge-Ring	Location	Near-Full	Random	Scratch	None
Center	**1632**	13	2	0	21	0	1	3	46
Donut	2	**187**	2	0	6	0	6	7	12
Edge-Loc	5	1	**1664**	16	67	2	18	26	277
Edge-Ring	3	0	145	3566	10	0	13	9	126
Location	31	18	58	1	**1036**	0	7	72	215
Near-Full	0	2	1	0	0	**52**	4	0	1
Random	4	7	29	0	25	10	**226**	3	43
Scratch	0	0	5	0	67	0	0	**335**	71
None	539	4	667	10	479	0	2	252	**57019**

B. Experimental Results and Analyses

We compare our proposed classifier against one classical machine learning techniques-based work [1] (refers to "TSM'15") and one cutting-edge convolutional neural network-based art [7] (denoted as "DAC'20") with the same labeled training and testing dataset in TABLE I. In [1], Radon-based features and geometry features are used in a support vector machine classification framework. Additionally, domain experts' intervention is used to relabel misclassified support vectors to further improve the accuracy. In [7], deep selective learning is exploited with distinct coverage on testing dataset for the defect pattern classification. In the absence of such expertise and to ensure fair comparison with our proposed approach, all methods are implemented without human intervention. To validate the proposed classifier, we first compare the classification performance of our model to those of "TSM'15" [1] and [7] under full coverage setting. TABLE II records the results of three methods with precision, recall and F_1 score and their corresponding macro-averaged values (i.e. arithmetic means). We can see that our method macro-averagely outperforms TSM'15 with 25.9% and 37.5% improvement on precision and recall rate and 43.0% rise on F_1 score. Moreover, it surpasses DAC'20 with a macro-averaged precision, recall and F_1 score of 11.5%, 21.5% and 20.1%, respectively. Overall, our approach achieves 82.5% macro-averaged recall rate (shown in TABLE II), which exceeds those of the counterparts (i.e. 60.0%



(a) TSM'15 [1]

(b) DAC'20 [7]

(c) Ours

Fig. 6 The heat maps of normalized confusion matrixes of three algorithms. The labels on x-axis represent the predicted wafer defect categories, while the annotations on y-axis refer to the real wafer defect classes. The diagonal elements are the recall rates. The unit for each element in the cell of the heat map is %.

of TSM'15 and 67.9% of DAC'20). It also behaves better on the actual failure pattern inspection (excluding None pattern) where

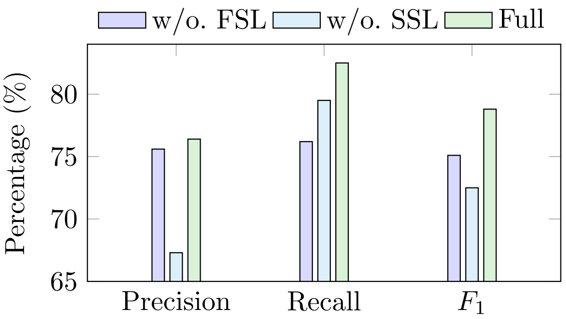


Fig. 7 The comparisons among different configurations on macro-averaged precision, recall and F_1 score.

80.7% macro-averaged recall rate for defect categories is attained compared to 64.8% for TSM'15 and 55.7% for DAC'20.

For a detailed demonstration, the confusion matrices of three works on the testing data are reported in Tables III to V. As a visual supplementary, the heat maps of associated normalized confusion matrixes of three algorithms are illustrated in Fig. 6. The diagram suggests that the diagonal cells of our algorithm have much darker colors than those of existing flows, whilst the off-diagonal elements of the proposed classification flow are light-colored. The above experimental results comprehensively demonstrate the superiority of our wafer failure pattern classification flow against the state-of-the-art works.

We also investigate how different configurations affect the macro-averaged classification performance. Fig. 7 outlines the contributions of few-shot learning and self-supervised learning paradigm to our flow. "w/o. FSL" refers to the flow trained with a typical cross-entropy loss module as a replacement to the few-shot learning loss, and "w/o. SSL" stands for the flow trained without self-supervised learning loss, while "Full" is our wafer failure pattern classifier with entire techniques. The histogram depicts that with the few-shot learning loss be replaced, the precision is slightly affected, while a noticeable drop emerges on the recall rate as well as on the F_1 score. We have good reason to believe that some minority failure pattern classes are poorly classified. As the bar graph expresses, without trained by self-supervised learning loss, each macro-averaged metric suffers a considerable decline. The result supports the argument that the classification flow gains from the self-supervised learning.

V. Conclusion

In this paper, we have proposed an end-to-end, CNN-based wafer failure pattern classification framework where incorporates the few-shot learning and self-supervised learning methods. It alleviates the imbalanced distribution issue in real industrial benchmark and takes full advantage of the massive unlabeled wafer data. Experimental results demonstrate that with surpassing a well-known machine learning-based method and a cutting-edge deep learning-based classification flow, our classifier has the potential to be a feasible alternative to manual inspection. In the future, we expect to apply and extend our flow to more complicated wafer inspection problems like mixed wafer failure pattern classification.

Acknowledgment

This work is partially supported by National Key R&D Program of China 2020YFA0711900, 2020YFA0711903, and The Research Grants Council of Hong Kong SAR CUHK14209420, CUHK14208021.

References

[1] 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 Transactions on Semiconductor Manufacturing (TSM)*, vol. 28, no. 1, pp. 1–12, 2015.

[2] J. Yu and X. Lu, "Wafer map defect detection and recognition using joint local and nonlocal linear discriminant analysis," *IEEE Transactions on Semiconductor Manufacturing (TSM)*, vol. 29, no. 1, pp. 33–43, 2016.

[3] M. B. Alawieh, F. Wang, and X. Li, "Identifying wafer-level systematic failure patterns via unsupervised learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 37, no. 4, pp. 832–844, 2017.

[4] T. Nakazawa and D. V. Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network," *IEEE Transactions on Semiconductor Manufacturing (TSM)*, vol. 31, no. 2, pp. 309–314, 2018.

[5] N. Yu, Q. Xu, and H. Wang, "Wafer defect pattern recognition and analysis based on convolutional neural network," *IEEE Transactions on Semiconductor Manufacturing (TSM)*, vol. 32, no. 4, pp. 566–573, 2019.

[6] T. Nakazawa and D. V. Kulkarni, "Anomaly detection and segmentation for wafer defect patterns using deep convolutional encoder–decoder neural network architectures in semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing (TSM)*, vol. 32, no. 2, pp. 250–256, 2019.

[7] M. B. Alawieh, D. Boning, and D. Z. Pan, "Wafer map defect patterns classification using deep selective learning," in *ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

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

[9] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Computing Surveys (CSUR)*, vol. 53, no. 3, pp. 1–34, 2020.

[10] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 2229–2238.

[11] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning (ICML)*, 2017, pp. 1126–1135.

[12] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," 2017.

[13] L. Bertinetto, J. F. Henriques, P. H. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," *International Conference on Learning Representations (ICLR)*, 2019.

[14] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 10 657–10 665.

[15] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2016, pp. 3637–3645.

[16] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4080–4090.

[17] H. Qi, M. Brown, and D. G. Lowe, "Low-shot learning with imprinted weights," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5822–5830.

[18] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, 2020, pp. 1597–1607.

[19] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9729–9738.

[20] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," *arXiv preprint*, 2020.

[21] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arXiv preprint*, 2020.

[22] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," *arXiv preprint*, 2021.

[23] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*, 2015, pp. 1–5.

[24] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu, "Hotspot detection via attention-based deep layout metric learning," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–8.

[25] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[26] A. Maurer, M. Pontil, and B. Romera-Paredes, "The benefit of multitask representation learning," *Journal of Machine Learning Research (JMLR)*, vol. 17, no. 81, pp. 1–32, 2016.

[27] Z. Ren and Y. J. Lee, "Cross-domain self-supervised multi-task feature learning using synthetic imagery," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 762–771.

[28] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean *et al.*, "TensorFlow: A system for large-scale machine learning," in *USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.