Wafer Map Defect Patterns Classification using Deep Selective Learning

Mohamed Baker Alawieh[1], Duane Boning[2], and David Z. Pan[1]

[1]ECE Department, University of Texas at Austin, Austin, TX, USA
[2]EECS Department, Massachusetts Institute of Technology, Cambridge, MA, USA

***Abstract*— With the continuous drive toward integrated circuits scaling, efficient yield analysis is becoming more crucial yet more challenging. In this paper, we propose a novel methodology for wafer map defect pattern classification using deep selective learning. Our proposed approach features an integrated reject option where the model chooses to abstain from predicting a class label when misclassification risk is high. Thus, providing a trade-off between prediction coverage and misclassification risk. This selective learning scheme allows for new defect class detection, concept shift detection, and resource allocation. Besides, and to address the class imbalance problem in the wafer map classification, we propose a data augmentation framework built around a convolutional auto-encoder model for synthetic sample generation. The efficacy of our proposed approach is demonstrated on the WM-811k industrial dataset where it achieves 94% accuracy under full coverage and 99% with selective learning while successfully detecting new defect types.**

I. Introduction

The continuous scaling of integrated circuit (IC) technologies along with the increase in state-of-the-art design complexity have exacerbated the challenges associated with designing robust circuits [1]. With such scaling, catastrophic defects and process variations stand out among the most prominent factors limiting the product yield of IC designs [1].

A critical first step towards improving yield during the IC design cycle is to identify the underlying factors that contribute most to yield loss, and for that, wafer map analysis is a key. Traditionally, wafer inspection was performed by experienced engineers who can identify the failure cause based on the wafer defect pattern. However, such process is tedious and an automated alternative is desired [2, 3].

In literature, different approaches have been proposed to address this task. In particular, machine learning techniques have been proposed to tackle the job using both unsupervised and supervised learning paradigms [2–8]. With unsupervised learning, clusters of wafer maps are constructed, and experienced engineers then label each of them with its defect pattern [3–5, 9]. however, defect labeling for wafers is not straightforward with such approach and it typically requires human experience. On the other hand, supervised learning techniques rely on features extracted from the wafer maps to build a classification model that is capable of classifying new wafer maps based on their defect type [2, 6, 7]. In [6, 7], features extracted based on spatial signature are used in a K-nearest neighbor framework for defect classification. Similarly, Radon-based features and a set of geometry features were used in a support vector machine framework for defect type classification in [2].

The aforementioned approaches rely on a set of features to capture the spatial properties of wafers. However, these wafers can be instinctively perceived as images with defect patterns being spatial features of these images. Hence, the native spatial characteristics of the defect patterns can be best preserved by using the natural representation of wafers as images. Recently, machine learning has achieved immense success in vision related tasks spanning different applications. Motivated by this success, we propose using convolutional neural networks to address the wafer map defect classification task as an image classification problem.

In this paper, we propose a novel framework for wafer map defect pattern classification using deep selective learning. Beside achieving superior accuracy compared to conventional approaches by leveraging the intrinsic image representation of a wafer, our proposed approach exhibits unique features that are tailored to address two challenges accompanying the task. One major challenge arises from the fact that some wafers may exhibit new defect patterns that were not previously seen by the model during training. In such a case, the model is expected to give a wrong label which can mask a new type of defects. Moreover, some wafer maps may exhibit more than one defect pattern which can overwhelm the classification model. To handle these cases, we propose using a convolutional neural network with an integrated *reject option* [10, 11]. In other words, given a user set compromise between risk and coverage during training, the model is trained to optimize for classification and rejection simultaneously. With this option, the model can choose to discard predictions with high risk of misclassification; i.e, the model abstains from prediction for some samples to maintain a low risk level. Clearly, the reject option can further improve the accuracy of the model on the selected samples. It can also be used to detect new defect classes and changes in the data distribution. In addition, the wafers not labeled by the model are expected to be *different*, i.e., these are wafers which engineers are interested in examining. Hence, the model can help allocate human resources where they are needed.

Secondly, defect classes have different frequencies of occurrence which typically result in an imbalanced training process where some minority classes are dominated by other majority ones. In this work, we propose using data augmentation to generate synthetic samples from the under-represented classes. In particular, we train a convolutional auto-encoder to generate samples from the distribution of the target class and use synthetic samples alongside the original ones in the training process [12].

With its unique features, the proposed approach can achieve superior results when compared to state-of-the-art wafer map defect classification approach when tested on the WM-811k industrial wafer map dataset [13]. Our main contributions can be summarized as follows:

- A novel approach for wafer map defect classification using convolutional neural networks is proposed.
- The proposed approach is equipped with an integrated reject option which can be leveraged to reduce the misclassification risk for the model.
- The selective learning feature can be used for new defect detection, data change detection, and resource allocation.

- A data augmentation framework based on convolutional auto-encoder is proposed to generate synthetic samples for under-represented defect classes.
- Experimental results demonstrate our proposed approach is able to achieve superior accuracy compared to the conventional approach with accuracy of 99% and 94% for the scenarios with and without the reject option.

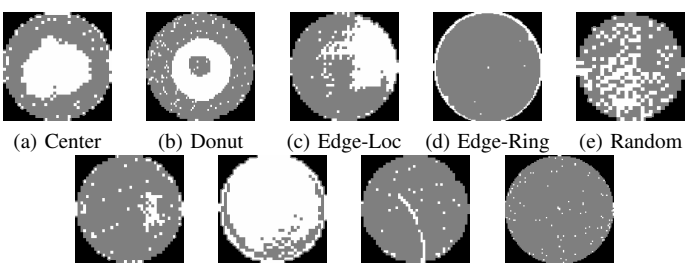
The rest of this paper is organized as follows. Section II presents the problem formulation and the used dataset. Section III provides a detailed explanation of the proposed approach. Section IV demonstrates the effectiveness of our approach with comprehensive results and applications, and conclusions are drawn in Section V.

II. Preliminaries

Identifying root causes behind IC defects is instrumental in maintaining a high yield. Towards this end, wafer defect classification plays an important role in correlating defect patterns with defect causes; and thus, helps address them in a timely manner to boost the yield.

In this work, we consider the WM-811k industrial wafer map dataset [13] which contains nine different wafer patterns. These patterns are obtained by marking die locations on the wafer with either *pass* or *fail*. Fig. 1 shows a sample wafer map from each defect type: *Center, Donut, Edge-Location, Edge-Ring, Random, Location, Near-Full, Scratch* and *None*. As shown in Fig. 1, the wafer maps are represented using a single channel grey-scale image of size 256×256 pixels with 3 pixel levels: 0, 127 and 255. Locations with pixel level 0 (i.e., black pixels in Fig. 1) are those not part of the wafer. Grey pixels with pixel level 127 represent die locations with a *pass* label while white pixels represent those with a *fail* label.

The objective is to train a model capable of classifying wafers into their corresponding defect type. Among works that have used the WM-811k industrial dataset, the state-of-the-art approach relies on a set of features extracted from the wafer maps combined with a support vector machine (SVM) classifier [2]. In such an approach, the feature extraction step can jeopardize some important information in the wafer map. Intuitively, the wafer maps can be viewed as images and the defect patterns are spatial features of these images. Hence, using the wafer maps in their image representation can best preserve the defect information. Thus, in this work, we propose a wafer map defect detection framework using deep selective learning featuring a convolutional neural network with an integrated reject option. In addition, we propose using a data augmentation scheme built around a convolutional auto-encoder to address the imbalance challenge in the wafer classification task.



(a) Center (b) Donut (c) Edge-Loc (d) Edge-Ring (e) Random

(f) Location (g) Near-Full (h) Scratch (i) None

Fig. 1: Sample wafer examples for different pattern types.

III. Wafer Map Defect Classification

In their nature, defect patterns are spatial patterns that can be visually detected on a wafer map image. Therefore, a suitable approach to classify different spatial patterns is through casting the problem as an image classification task and leveraging convolutional neural networks to handle it. In this work, we propose using a convolutional neural network equipped with (i) a reject option and (ii) a convolutional auto-encoder framework for data augmentation. The details of these two features will be discussed in sections III-A and III-B respectively.

For the core classification task, we develop a convolutional neural network (CNN) with 3 convolutional layers and one fully-connected layer (FC) as shown in Table I. All three convolutional layers are associated with a 2×2 maximum pooling step. The output layer is also a fully-connected layer with a number of neurons equal to the number of target classes n_c. To get the prediction, a softmax operator is applied to the output layer and a one-hot vector is generated such that the index of the class with the highest softmax score is set to 1. The typical loss function used for such problem is the cross-entropy function which can be expressed, for a single sample, as [14]:

$$l(f(x_i), y_i) = \sum_{k=1}^{n_c} y_i^{(k)} \log f^{(k)}(x_i), \tag{1}$$

where x_i is a sample input image, y_i is a one-hot vector of size n_c whose $k-$th element $y_i^{(k)}$ is equal to one if k is the true class label of sample x_i, and $f(x)$ represents the CNN model whose output is the one-hot encoded label prediction.

Layer Name	Conv1	Conv2	Conv3	FC	Output
# of Filters	64	32	32	256	n_c
Filter Size	5x5	3x3	3x3	-	-
Pooling	2x2	2x2	2x2	-	-

TABLE I: The details of the core CNN architecture.

A. Selective Learning

Machine learning model trust and risk analysis have been recently under spotlight [15, 16]. The questions of when to trust a particular machine learning prediction and how to quantify the risk associated with acting upon it are still not fully answered. In our work, we equip the CNN model with a reject option which allows the model to abstain from making predictions for particular samples when the risk of misprediction is high [10, 11]. This option provides a compromise between risk and coverage, where coverage is a measure that reflects the probability of the model **not** abstaining from making predictions. In other words, if the cost associated with a misprediction is high, the user may choose to reduce the risk at the cost of reducing the coverage as well.

This option comes in handy when tackling the wafer map defect classification problem since the model can abstain from making predictions in two ambiguous scenarios. The first is when a new defect pattern, not seen during training, appears on a wafer, while the second is when more than one known pattern appear on the wafer. The aim behind using the reject option is to push the model to abstain from making predictions in these two scenarios in addition to any other scenario where the prediction is believed to be risky. Further details about the applicability of such model are shown in Section IV-D.

A selective model is a pair (f, g), where f is the prediction function, and g is a selection function which servers as binary qualifier for f [10, 17]:

$$(f, g)(x_i) = \begin{cases} f(x_i), & \text{if } g(x_i) = 1 \\ \text{abstain}, & \text{if } g(x_i) = 0 \end{cases}. \tag{2}$$

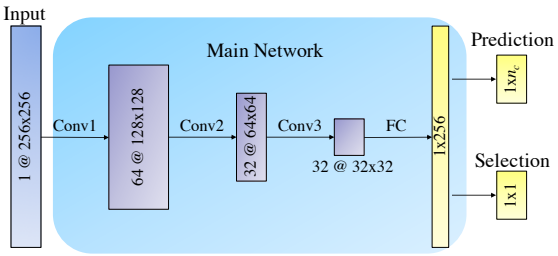


Fig. 2: The CNN network architecture showing both the prediction and selection heads.

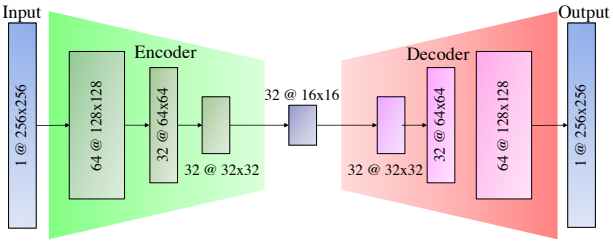


Fig. 3: The auto-encoder network architecture showing both the encoder, decoder and the latent space representation z.

Hence, the selective model chooses to abstain from prediction when $g(x_i) = 0$ which represents a trade-off between risk and coverage. In this context, coverage is defined as[10]:

$$C(g) = \mathbf{E}[g(x)], \tag{3}$$

which represents the probability mass of the unrejected region.

As for the misclassification risk, in the case of a model with f alone, it can be expressed as:

$$R(f) = \mathbf{E}[l(f(x_i), y_i)], \tag{4}$$

however, the risk in the selective model (f, g) is given by [10]:

$$R(f, g) = \frac{\mathbf{E}[l(f(x_i), y_i)g(x_i)]}{C(g)}. \tag{5}$$

Given labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, the empirical estimations of the expressions in (3) and (5) can be expressed as:

$$c(g|\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} g(x), \tag{6}$$

$$r(f, g|\mathcal{D}) = \frac{\frac{1}{N}\sum_{i=1}^{N} l(f(x_i), y_i)g(x_i)}{c(g)}. \tag{7}$$

Based on equation (7), one can clearly notice that risk can be traded-off for coverage in this setup. To define an optimal selective model, we elect to minimize the selective risk given a constraint on the coverage. In the scope of the CNN model proposed earlier, this requires updating the loss function to include the selective risk, and the model architecture to predict g in addition to f [10]. To do so, we first update the network architecture to include two output heads. The first is the prediction head implementing the main function f, while the other is a selection head consisting of a single neuron with a sigmoid activation to implement the function g. These two heads depart at the end of the network architecture after the main blocks including convolutional fully connected layers as shown in Fig. 2.

The new objective of the training process is to minimize the selective loss while meeting the coverage constraints. This can be expressed mathematically as new loss function [10, 18]:

$$\begin{aligned} \mathcal{L}_{(f,g)} &= r(f, g|\mathcal{D}) + \lambda\Psi(c_0 - c(g|\mathcal{D})) \\ \text{where} \quad \Psi(z) &= \max(0, z)^2. \end{aligned} \tag{8}$$

Here, c_0 is the target coverage, λ is hyper-parameter reflecting the importance of the coverage constraint, and Ψ is a quadratic penalty function.

The overall training objective is two minimize a weighted combination of the selective loss in (8) and the empirical loss of f obtained from (4) which is expressed as [10]:

$$\mathcal{L} = \alpha\mathcal{L}_{(f,g)} + (1 - \alpha)r(f|\mathcal{D}), \tag{9}$$

where α is a hyper-parameter controlling the importance of loss terms. The use of the empirical loss of f alongside the selective loss is essential because, in its absence, the network will focus on a fraction c_0 of the dataset and overfit to a subset of the training data. With this term added, the network is exposed to all training instances throughout the training process [10].

B. Data Augmentation

In many classification tasks, class imbalance shows up as a critical challenge that can hinder the performance of classifiers [19, 20] which necessitates addressing it during data preparation. In the wafer defect classification problem, different patterns have different probabilities of occurrence which results in an imbalance issue that should be addressed in the pre-processing phase.

In this work, we elect to use data augmentation to address the imbalance problem [19, 21]. The key idea is to generate synthetic samples for the under-represented classes to augment the original dataset and reduce the effect of class imbalance. The first step is to learn the underlying distribution of the under-represented classes so that new examples can be sampled from the learned distribution. This can be done using auto-encoder networks trained to learn a latent space representation of the samples in the target class through an encoding-decoding scheme [12, 22, 23]. After training, the encoding block can generate latent space representations of the original images which, when passed through the decoding block, are expected to re-generate the original images with high accuracy. However, if small perturbation is added to the latent space representations before the decoding stage, new synthetic samples, that are close to the original ones, can be generated and used to augment the dataset. Since the samples in the problem at hand are images, we propose using a convolutional auto-encoder as the core engine in our data augmentation framework.

As a first step we train a convolutional auto-encoder for an under-represented class cl with the objective of reconstructing the samples corresponding to cl in the training dataset. Fig. 3 shows the architecture of the convolutional auto-encoder consisting of the encoder, decoder, and the latent space representation z forming the network bottleneck. For the encoder network, the number of filters in each convolutional stage is shown in Fig. 3 with all filters of size 5×5 and a 2×2 maxpooling operation accompanying each layer. The decoder network is a mirrored version of the encoder with the same architecture having deconvolution and upsampling replacing the convolution and maxpooling operations in the encoder respectively. More details about training the network are omitted due to page limit. Reader is referred to [12] for more details.

In addition to using the auto-encoder for generating synthetic images, image rotation is also employed. Given a target minimum number of samples required for each class T and a set of n_{cl} original samples $\mathcal{P}$ belonging to class cl, the data augmentation process is executed as summarized in algorithm 1.

As a first step after training the convolutional auto-encoder, the number of required rotations per sample necessary to meet

the target T is computed (line 1). Next, for each image in $\mathcal{P}$, the latent space representation z (line 3) is obtained. Then, for each rotation angle, z is perturbed by adding Gaussian noise with zero mean and small standard deviation σ_0 (line 5) before passing it to the decoder to generate a new image (line 6). The resultant image is expected to have a continuous spectrum of pixels and not only the desired three levels $0, 127$ and 255; hence, a quantization step is applied to map pixels to these three levels (line 7). The generated image is then rotated (line 8) and salt and pepper (s&p) noise is added to the image (line 9) before adding it to the set of synthetic images Ω. Here, s&p noise is added by randomly selecting few die locations and flipping their label; i.e, switch a *pass* to *fail* and vice versa.

Algorithm 1 Data augmentation for an underrepresented class

Require: A pool of data samples $\mathcal{P}$, T, n_{cl}
1: Train convolutional auto-encoder for cl; $\Omega \leftarrow \emptyset$; $n_r = \lceil T/n_{cl} \rceil - 1$;
2: **for** img in $\mathcal{P}$ **do**
3: Get z, the latent space representation of img using the encoder;
4: **for** i in $\{0, \ldots, n_r - 1\}$ **do**
5: $z' \leftarrow z + \mathcal{N}(0, \sigma_0^2)$;
6: Pass z' through decoder to get image img';
7: Quantize img' to desired 3 level mapping;
8: Rotate img' by angle $i \times 360/n_r$;
9: Add s&p noise to the image;
10: $\Omega \leftarrow \Omega \cup img'$;
11: **end for**
12: **end for**
13: **return** Ω.

After obtaining the set of synthetic samples Ω using algorithm 1, it can be merged with the original dataset during training. If a direct merge of Ω and $\mathcal{P}$ is done, original and synthetic samples will have equal weight during the training process. Here, we introduce a weight hyper-parameter $w < 1$ to multiply the loss terms corresponding to Ω in the objective function. Intuitively, this translates to penalizing the objective function $1/w$ more when an original sample is misclassified compared to when a synthetic sample is.

IV. Experimental Results

A. Dataset

In our experiments, we use a labeled subset of WM-811k industrial wafer dataset [13] containing 54355 wafer maps from the different wafer classes as listed in the Dataset section of Table II where all the wafer maps are scaled to 256×256 pixel images.

It is important to note that the test-train split used in our work differs from that used in [2] for few reasons. First, the test set used in [2] is dominated by the *None* class due to the inherent data imbalance. While this is natural in a usage scenario, the overall accuracy is not a representative metric under this testing setup. A model that is capable of learning this majority class can achieve very high overall accuracy, while still performing poorly on the defect classes which are more important for yield analysis. In fact, when the *None* class is excluded from the analysis, the method proposed in [2] achieves 44.35% accuracy on the test set.

Another observation is that the defect data in the "Train" and "Test" sets used in [2] are significantly different. One indication of this is the low accuracy [2] achieves when *None* class is excluded. To further validate this in our experiments, the "Train" data was split into 0.7:0.1:0.2 for training, validation and test. In other words, 20% of the "Train" data is kept for testing while 80% is used for training/validation. Results showed that the model, under full coverage, can achieve 97%, 94% and 94% accuracy on the three splits respectively; however, it performs poorly on the "Test" set. While generalization to the unseen samples from the "Train" set shows that no over-fitting is taking place, poor performance on the "Test" set suggests data discrepancy. Another indication for this is that, under a selective labeling scenario with 50% coverage as the minimum target, the results on the 3 splits in the "Train" data achieved 99% accuracy with actual coverage between 45-57%. However, coverage on the "Test" data was about 5% while still achieving 99% accuracy on the selected samples.

On one hand, these experiments imply that there exists major differences in the distributions of the original "Train" and "Test" data. On the other hand, it illustrates the capability of our selective learning approach to detect the change in the distribution of data when the actual coverage significantly drops below the target one.

Therefore, and to properly demonstrate the usability of selective learning, we elect to use a coherent dataset for the results shown in the remaining of this section through using the original "Train" set only and splitting it into 0.8:0.2 train-test split denoted by "Training" and "Testing" in Table II.

B. Data Augmentation

By examining the data distribution in Table II, one can notice that class *None* dominates all other defect classes. This reflects the class imbalance issue in the dataset. Therefore, it is important to address this issue using data augmentation in a pre-processing step. In practice, the accuracy on defect classes is more important for yield analysis compared to that of the *None* class. Hence, for each of the defect classes, data augmentation is performed according to the procedure in algorithm 1 with the target number of samples T set to 8000. Column Train$_{aug}$ in Table II shows the number of samples for each class after the augmentation. Here, data augmentation is performed on the training data only. This is mainly because we are interested in improving the training process while maintaining valid testing on original data only without including synthetic data in the test set. Moreover, when augmenting each class, only training samples from the class are used in algorithm 1, in other words, testing samples are left out in both data augmentation and model training stages.

Fig. 4 illustrates a sample result from the data augmentation process. The first row in Fig. 4 represents sample original images

	Dataset			c_0=0.2				c_0=0.5				c_0=0.75			
	Training	Testing	Train$_{aug}$	Prec	Rec	f_1	Cov	Pre	Rec	f_1	Cov	Pre	Rec	f_1	Cov
Center	2767	695	8308	0	0	0	2	0.99	0.96	0.97	268	0.99	0.97	0.98	634
Donut	329	80	8180	0.9	1	0.95	9	0.8	0.94	0.86	17	0.93	0.77	0.84	56
Edge-Loc	1958	459	9668	0	0	0	6	0.57	0.24	0.33	17	0.75	0.49	0.59	132
Edge-Ring	6802	1752	10686	1	0.93	0.96	27	1	1	1	1370	1	0.98	0.99	1682
Location	1311	309	11664	0.93	0.84	0.88	61	0.57	0.19	0.29	21	0.46	0.59	0.52	157
Near-Full	49	5	8164	0	0	0	1	1	0.6	0.75	5	1	0.5	0.67	4
Random	498	111	8282	-	-	-	0	0	0	0	1	0.6	0.19	0.29	16
Scratch	413	87	8400	0	0	0	5	0	0	0	4	0.09	0.15	0.11	27
None	29357	7373	29357	0.99	1	1	2853	0.99	1	0.99	3947	0.98	0.99	0.98	6981
Overall	43484	10871	102697	Accuracy = 99.1%			2964 (27.2%)	Accuracy = 99.0%			5650 (57.9%)	Accuracy = 96.6%			9689 (89.1%)

TABLE II: Details of the dataset and the results of selective learning under different coverage setup.

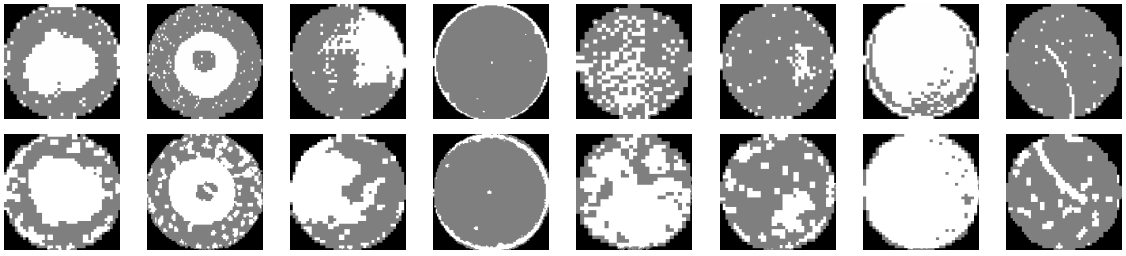


Fig. 4: Sample data augmentation results where original (synthetic) samples are shown in the first (second) row respectively.

from the defect classes while the second row shows corresponding sample synthetic images generated using algorithm 1.

C. Classification Performance

With the imbalance issue handled using data augmentation, the convolutional neural network presented in Fig. 2 is trained for 100 epochs with Adam optimizer. Both hyper-parameters λ in (8) and α in (9) are set to 0.5 throughout our experiments. On the other hand, the parameter c_0 in (8), representing the minimum required coverage, is changed throughout the experiments to demonstrate the risk versus coverage compromise.

Our proposed approach is compared against the work in [2] (denoted by SVM) to address classifying wafer maps in the WM-811k dataset . In [2], Radon-based features and geometry features are used in a support vector machine classification framework. In addition, domain experts intervention is used to relabel misclassified support vectors to further improve the accuracy. In this work, in the absence of such expertise and to ensure fair comparison with our proposed approach, both methods are implemented without human intervention.

To validate our model, we first compare the performance of our deep learning model under full coverage to that of SVM [2]. Table III shows the resulting confusion matrices corresponding to the testing data. Overall, our approach achieves 94% accuracy compared to 91% for SVM, while also performing better on the actual defect classes (excluding *None*) where our model achieves 86% correct detection rate for defect classes compared to 72% for [2]. While this shows the competitiveness of our deep learning model, our key feature is actually the selective learning framework governing the trade-off between risk and coverage.

To demonstrate this trade-off, which is the key idea selective learning is based upon, we vary the coverage requirement value, c_0 in (9) to take the values $[0.2, 0.5, 0.75, 1]$. For the case when $c_0 = 1$, we train the model with cross-entropy loss function only and report accuracy for the entire test set, hence, the test coverage is 1 (results for this setup are shown in Table III). As one would expect, demanding higher coverage is usually accompanied with some sacrifice in accuracy. This trend is demonstrated in Fig. 5 showing the evaluation of the accuracy and coverage metrics for the different values of c_0, where a clear trade-off is shown between coverage and risk (or accuracy) in which higher coverage results in higher misclassification risk.

For different c_0 values, Table II summarizes the results where four metrics are reported for each class: precision (Pre), recall (Rec), f_1-score (f_1), and actual coverage (Cov). Due to space limit, we elect to show the these metrics instead of the confusion matrices since they reflect the model performance per class. As for the actual coverage (Cov), it represents the number of samples from each class the model chooses to label. Also shown in the last row of the table are overall accuracy and total coverage values. Three observations can be made based on these results.

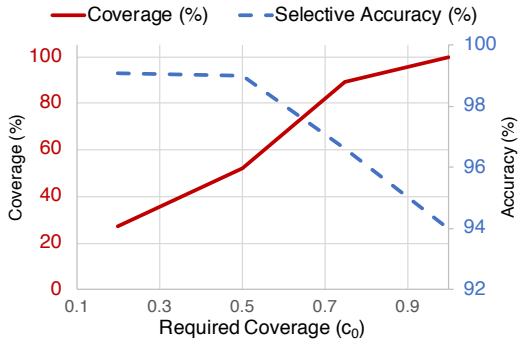


Fig. 5: The selective accuracy and test coverage plotted as a function of c_0 shows the trade-off between risk and coverage.

First, the model is indeed choosing the samples with low risk to label which is clear from the overall accuracy values. Besides, examining per group performance validates this point since the model has higher coverage for classes with high f_1 scores as in the case of classes *Center*, *Edge-Ring* and *None* for $c_0 = 0.5$. Another observation is that, for all cases, the actual coverage on the test set is higher than the minimum required coverage c_0. Finally, for the case of $c_0 = 0.75$, the model is able to achieve 96.6% accuracy with around 90% coverage which is 2.6% higher than the 94% accuracy achieved under full coverage, and the 91% achieved by [2]. This shows that, even under high coverage demand, the selective learning scheme is still able to minimize misclassification risk.

D. Model Usability

The proposed selective learning scheme for wafer map defect detection has many advantages on the application side. We list here three of these applications: (i) detection of new defect class(es), (ii) resource allocation for human in the loop setup, and (iii) detection of changes in the data distribution.

One major advantage of the selective learning scheme is that it allows detecting a new defect class when it shows up. Intuitively, if a new defect occurs, the model should abstain from labeling the new defect samples because they are associated with high risk. To validate this utility, we look into the class recall for an experiment with 50% coverage shown in Table IV, where class *Near-Full* was excluded from the training process and all its samples were used during testing. This is done to test whether selective learning will label the samples from the unseen class. As shown in the table (highlighted with light blue), the original recall (neglecting the reject option) is 0% since the model has to give one of the 8 available labels to the new samples. However, with selective learning, the model abstained from predicting a label for all samples belonging to the new class.

Another application is for resource allocation. Such a model is developed to reduce the cost associated with having experienced

Proposed	*Center*	*Donut*	*Edge-Loc*	*Edge-Ring*	*Location*	*Near-Full*	*Random*	*Scratch*	*None*
Center	**665**	2	1	0	11	0	0	3	13
Donut	0	**58**	1	0	15	0	0	3	3
Edge-Loc	1	0	**321**	5	51	0	0	16	65
Edge-Ring	0	0	44	**1683**	13	0	0	7	5
Location	4	2	13	0	**199**	0	1	18	72
Near-Full	0	0	1	0	0	**2**	2	0	0
Random	2	2	16	1	14	0	**63**	2	11
Scratch	1	0	4	0	38	0	0	**25**	19
None	1	0	8	0	113	0	0	48	**7203**

SVM [2]	*Center*	*Donut*	*Edge-Loc*	*Edge-Ring*	*Location*	*Near-Full*	*Random*	*Scratch*	*None*
Center	**552**	2	9	1	1	1	0	0	129
Donut	12	**29**	20	2	1	12	0	0	4
Edge-Loc	25	0	**231**	52	4	15	0	0	132
Edge-Ring	2	0	2	**1678**	0	0	0	0	70
Location	65	2	78	1	**5**	0	0	0	158
Near-Full	1	0	0	0	0	**4**	0	0	0
Random	0	1	3	26	0	0	**0**	0	81
Scratch	32	0	4	0	4	14	0	**33**	0
None	3	0	3	10	0	0	0	0	**7357**

TABLE III: The confusion matrices for our proposed approach (under full coverage), achieving 94% accuracy, and SVM [2] achieving 91% accuracy are shown. Our approach demonstrates better accuracy detecting actual defect classes compared to [2].

	Original Recall	Selective Recall	Coverage
Center	0.96	0.93	129 (18.5%)
Donut	0.89	1.00	20 (25.0%)
Edge-Loc	0.79	0.00	8 (1.7%)
Edge-Ring	0.98	0.99	1271 (72.5%)
Location	0.72	0.75	28 (9.1%)
Random	0.80	-	0 (0%)
Scratch	0.22	0.00	7 (8.05%)
None	0.99	0.67	3573 (48.4%)
Near-Full	0.00	-	0 (0%)

TABLE IV: Results of an experiment where *Near-Full* class was not included in training.

engineers manually label the wafer. However, the models are still being used under the supervision of those engineers. While the model is in use, there is typically some budget for manual examination by the engineers; however, determining the best small subset of wafers to thoroughly examine is not simple. With selective learning, coverage can be set such that the model labels the majority of samples while passing the ones with high risk for examination. This in fact provides a perfect allocation of resources as the model is predicting with high confidence and the high risk samples, which are typically the most interesting for the engineers, are automatically detected and passed for examination.

The third application of this scheme is detecting concept shifts or major changes in the distribution of the data. Under such scenario the actual coverage of the model would drop significantly; hence, raising a flag that the model needs to be retrained on a new dataset. This was encountered in our experiments as mentioned in Section IV-A.

V. Conclusion

In this work, we present a novel 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. This option can significantly increase the trust in the classification model and facilitate its wide adoption. Moreover, the proposed approach features a data-augmentation step using a convolutional auto-encoder to address the class imbalance. Experimental results demonstrate our approach can achieve superior accuracy when compared to conventional approach with 99% accuracy under selective learning framework and 94% under full coverage setting.

Acknowledgment

This work is supported in part by NSF under Award No. 1718570.

References

[1] *International Roadmap for Devices and Systems*. Semiconductor Industry Associate, 2016.

[2] 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*, vol. 28, no. 1, pp. 1–12, 2014.

[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*, vol. 37, no. 4, pp. 832–844, 2017.

[4] F.-L. Chen and S.-F. Liu, "A neural-network approach to recognize defect spatial pattern in semiconductor fabrication," *IEEE transactions on semiconductor manufacturing*, vol. 13, no. 3, pp. 366–373, 2000.

[5] C.-F. Chien, S.-C. Hsu, and Y.-J. Chen, "A system for online detection and classification of wafer bin map defect patterns for manufacturing intelligence," *International Journal of Production Research*, vol. 51, no. 8, pp. 2324–2338, 2013.

[6] K. W. Tobin Jr, S. S. Gleason, T. P. Karnowski, S. L. Cohen, and F. Lakhani, "Automatic classification of spatial signatures on semiconductor wafer maps," in *Metrology, Inspection, and Process Control for Microlithography XI*, vol. 3050. International Society for Optics and Photonics, 1997, pp. 434–444.

[7] T. P. Karnowski, K. W. Tobin Jr, S. S. Gleason, and F. Lakhani, "Application of spatial signature analysis to electrical test data: validation study," in *Metrology, Inspection, and Process Control for Microlithography XIII*, vol. 3677. International Society for Optics and Photonics, 1999, pp. 530–541.

[8] J. Y. Hwang and W. Kuo, "Model-based clustering for integrated circuit yield enhancement," *European Journal of Operational Research*, vol. 178, no. 1, pp. 143–153, 2007.

[9] M. B. Alawieh, F. Wang, and X. Li, "Identifying systematic spatial failure patterns through wafer clustering," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 910–913.

[10] Y. Geifman and R. El-Yaniv, "SelectiveNet: A deep neural network with an integrated reject option," in *International Conference on Machine Learning (ICML)*, 2019.

[11] C.-K. Chow, "An optimum character recognition system using decision functions," *IRE Transactions on Electronic Computers*, no. 4, pp. 247–254, 1957.

[12] D. Holden, J. Saito, T. Komura, and T. Joyce, "Learning motion manifolds with convolutional autoencoders," in *SIGGRAPH Asia 2015 Technical Briefs*. ACM, 2015, p. 18.

[13] "WM-811K wafer map," https://www.kaggle.com/qingyi/wm811k-wafer-map, accessed: 2019-07-30.

[14] R. Y. Rubinstein and D. P. Kroese, *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[15] W. Ye, M. B. Alawieh, M. Li, Y. Lin, and D. Z. Pan, "Litho-GPA: Gaussian process assurance for lithography hotspot detection," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*. IEEE, 2019, pp. 54–59.

[16] K. Lee, H. Lee, K. Lee, and J. Shin, "Training confidence-calibrated classifiers for detecting out-of-distribution samples," in *International Conference on Learning Representations (ICLR)*, 2018.

[17] R. El-Yaniv and Y. Wiener, "On the foundations of noise-free selective classification," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1605–1641, 2010.

[18] F. A. Potra and S. J. Wright, "Interior-point methods," *Journal of Computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 281–302, 2000.

[19] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intelligent data analysis*, vol. 6, no. 5, pp. 429–449, 2002.

[20] Y. Lin, M. B. Alawieh, W. Ye, and D. Z. Pan, "Machine learning for yield learning and optimization," in *IEEE International Test Conference (ITC)*, 2018.

[21] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: when to warp?" in *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE, 2016, pp. 1–6.

[22] M. B. Alawieh, Y. Lin, W. Ye, and D. Z. Pan, "Generative learning in VLSI design for manufacturability: Current status and future directions," *Journal of Microelectronic Manufacturing*, vol. 2, no. 5, pp. 1–12, 2019.

[23] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.