Advanced Outlier Detection Using Unsupervised Learning for Screening Potential Customer Returns

Hanbin Hu
Department of ECE
University of California
Santa Barbara, CA 93105
hanbinhu@ucsb.edu

Nguyen Nguyen
NXP Semiconductors
Austin, TX 78735
nguyen.nguyen@nxp.com

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

Peng Li
Department of ECE
University of California
Santa Barbara, CA 93105
lip@ucsb.edu

***Abstract*—Due to the extreme scarcity of customer failure data, it is challenging to reliably screen out those rare defects within a high-dimensional input feature space formed by the relevant parametric test measurements. In this paper, we study several unsupervised learning techniques based on six industrial test datasets, and propose to train a more robust unsupervised learning model by self-labeling the training data via a set of transformations. Using the labeled data we train a multi-class classifier through supervised training. The goodness of the multi-class classification decisions with respect to an unseen input data is used as a normality score to defect anomalies. Furthermore, we propose to use reversible information lossless transformations to retain the data information and boost the performance and robustness of the proposed self-labeling approach.**

***Keywords*—unsupervised learning, outlier detection, self-labeling, post-silicon testing.**

I. Introduction

Screening out all potentially defective parts before shipping to customers is crucial for minimizing the risk of the products failing in the customer line or field [1]. Typically, test process consists of wafer probe test, burn-in test with packaged parts, and final test, as shown in Fig. 1. During both the wafer probe and final test phases, a large number of parametric tests are performed to extract the part performance values. Outlier detection is applied using the results from the parametric tests to identify abnormal parts. Such parametric tests and outlier detection are especially important to test analog and mixed-signal circuits, as a defect in those circuits is more likely to cause a parametric shift rather than a hard functional failure.

The typical measure of design quality is DPPM, i.e., number of Defective Parts which fail after shipping to the customer (also known as customer failures) Per Million parts shipped. The target quality level for chips that are deployed in mission-critical applications, e.g. automotive electrics, can be very stringent. As the complexity of semiconductor products keeps increasing, it is increasingly challenging for post-silicon testing to screen out all defects without any escape to customers.

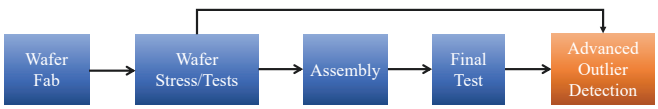
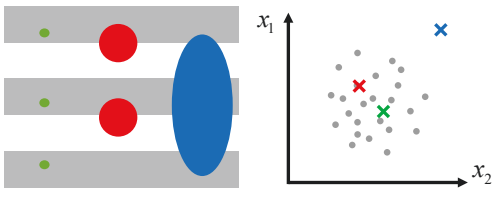


Fig. 1: Post-silicon production test flow.

While it is critical to learn from extremely rare (in DPPM level) customer failures to improve outlier defect detection to reach the Zero Defect (or zero customer failure) quality, there remain major challenges in this learning process. **1)** Customer failures are rare, as they are escapes from a rather comprehensive test process. As nearly-all of the defective parts have been screened out, data on customer failures are extremely scarce, typically at several parts per million (PPM) level at most. **2)** It can be difficult to identify a subset of parametric tests that expose potential failures. Alternatively, outlier detection may be performed over a high-dimensional input feature space formed by a large number of parametric test measurements. **3)** It is difficult to catch latent reliability faults by comparing with the normal chip data distribution, leading to defect escapes.

As one illustrative example, Fig. 2a shows different types of defects around parallel wires (in gray) on a particular metal layer, and Fig. 2b gives the corresponding distribution of two parametric test results. The small green defects have no impact on the circuit performance, which also locate in the center of the distribution of the parametric test data. The large blue defect can cause catastrophic short-circuit fault and are typically far away from the center of the distribution, making it easy to screen it out. The red defects, the so-called latent reliability defects, might not be caught by post-silicon testing, however, can evolve into early life failures in the customer field due to aging. Such latent defects are extremely hard to be detected unless a large combination of different parametric tests is analyzed in detail to distinguish them from normal circuit behaviors or inconsequential defects during post-silicon testing.



(a) Defect illustration (b) Defect distribution

Fig. 2: Illustration of potential defects types.

978-1-7281-9113-3/20/$31.00 ©2020 IEEE

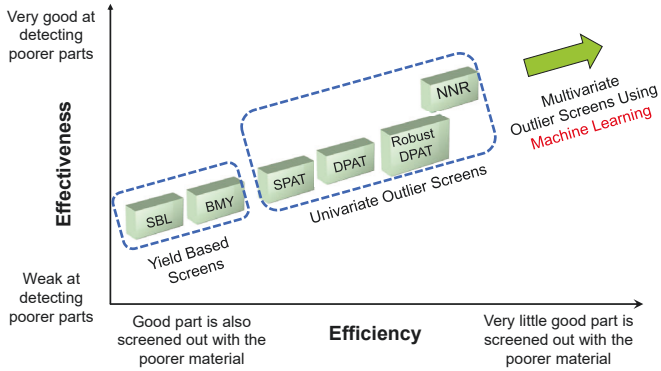


Fig. 3: Advanced outlier detection evolution.

Due to the scarcity of failure data, typically there are insufficient defective samples to validate *unsupervised* models for outlier detection [2]. As shown in Fig. 3, most early works in this field applied yield based screening strategies like statistical bin limits (SBL) and below minimum yield (BMY) [3], [4]. In order to more efficiently and effectively screen out outlier parts and reduce the corresponding yield loss, several univariate outlier detection methods such as static/dynamic part average test (S/DPAT) [5]–[8], and nearest neighbor residual (NNR) [9]–[11], and location average [9], [12] were proposed and are commonly employed in the industry. Multivariate outlier detection methods were also proposed for screening rare defects and customer returns [13], [14]. However, with recent advances in the machine learning field, it remains interesting to study how advanced machine learning techniques can be helpful for outlier detection in test.

This paper aims to study modern machine learning techniques on outlier detection in view of screening defect escapes to customers. The purposes of this paper are two-fold. First, we assess the application of the advancements in anomaly detection [15], [16] from the field of machine learning to the targeted testing problem and observe their limitations. We consider several popular unsupervised anomaly detection methods trained using normal data and proposed in machine learning. Tree-based methods such as isolation forest [17] flag a detected anomaly when the average path length to the leaves in the forest falls below a threshold. One-class support vector machine (OCSVM) [18]–[20] bounds the normal training data within a tight boundary, which is used to separate normal data from abnormal data. Autoencoders [21], [22] or generative adversarial networks (GAN) [23], [24] are among the most successful methods where any observed large reconstruction error signifies anomaly.

While demonstrating certain degrees of success in other anomaly detection problems, we show that the aforementioned methods do not work well for the challenging problem of identifying extremely-rare customer failures so as to minimize defect escape to customers. Hence, the second purpose of this paper is to bring a new perspective to post-silicon testing by adapting the geometric transformation based deep anomaly image detection approach [25], which leverages supervising learning for solving the unsupervised learning problem. More specifically, [25] creates a set of self-labeled images by transforming each example in the given raw training dataset using a number of geometric transformations. Each transformed image is labeled using the index of the transformation applied. A multi-class classifier is trained using the self-labeled training data. During inference, an unseen image is first transformed using the same set of geometric transformations. The resulting transformed images are classified by the trained multi-class classifier. The goodness of the classification decisions is considered as a normality score, which is used to signify detection of abnormality when the normality score drops to a low value. We introduce two key modifications to make this self-labeling approach viable for the intended extremely-rare customer failure detection problem. First, we replace geometric transformations used for images by nonlinear transformations suitable for processing test data. Second, we introduce a family of reversible information lossless transformations to boost the performance and robustness of the self-labeling methods. Experimentally, we demonstrate that the proposed self-labeling approach significantly outperforms the other methods in terms of prediction accuracy and robustness using a large set of public datasets and real industrial post-silicon test data.

II. Unsupervised Outlier Detection

A. Problem Formulation

Consider a D-dimensional input space $\mathcal{X} \subseteq \mathbb{R}^D$ containing all potential inputs, e.g. parametric post-silicon test results. Let $\mathcal{X}_N \subseteq \mathcal{X}$ and $\mathcal{X}_A \subseteq \mathcal{X}$ represent normal and abnormal inputs, e.g. the test results of good vs. failing (outlier) chips, with $\mathcal{X}_N \cap \mathcal{X}_A = \emptyset$ and $\mathcal{X}_N \cup \mathcal{X}_A = \mathcal{X}$. To classify an input $\mathbf{x} \in \mathcal{X}$ as normal or abnormal, an unsupervised learning method learns a binary classification function $f : \mathcal{X} \rightarrow \{0, 1\}$, where "0" indicates normality (true negative example), i.e., $\mathbf{x} \in \mathcal{X}_N$; "1" represents outlier (true positive example), i.e., $\mathbf{x} \in \mathcal{X}_A$.

Without labeled outlier data due to its scarcity, well developed supervised learning cannot be applied. Instead, as shown in Fig. 4, one learns a score function $s : \mathcal{X} \rightarrow \mathbb{R}$ through certain unsupervised learning algorithm to assess the normality of the seen normal data. During inference phase, the larger value of $s(\mathbf{x})$, the more likely the unseen data is normal. With a specified decision threshold s_{Th}, the binary classification model f trained without supervision is

$$f_{Th}(\mathbf{x}) = \begin{cases} 0 & s(\mathbf{x}) \geq s_{Th} \\ 1 & s(\mathbf{x}) < s_{Th} \end{cases} \quad (1)$$

Note that the input features may not fully reveal the outlier information in practice, implying $\mathcal{X}_N \cap \mathcal{X}_A \neq \emptyset$. However, the score function definition still works properly under this case, indicating how likely the given input x appears to be an outlier.

B. Performance Metric of Machine Learning Models

The choice of the decision threshold s_{Th} in (1) has a large impact on the quality of outlier detection. It is unfair to select specific thresholds when comparing different outlier

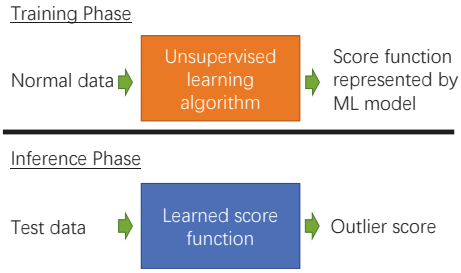
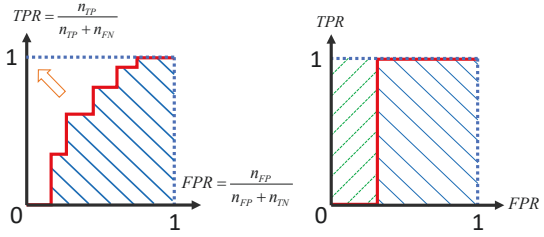


Fig. 4: Unsupervised outlier detection illustration.



(a) General case. (b) Single outlier case

Fig. 5: ROC curve and AUROC.

detection methods since the scale and the distribution of the score function value vary widely from methods to methods. We apply the widely adopted Area Under Receiver Operation Characteristic (ROC) curve (AUROC) to compare different models without relying on specific thresholds.

As shown in Fig. 5, the ROC curve characterizes the true positive rate (TPR) and false positive rate (FPR) of a model as the threshold s_{Th} is swept. If s_{Th} exceeds the maximum score function value of a testing dataset, all data will be considered as outliers, making both TPR and FPR 1.0; the other way around will make both TPR and FPR 0. The ROC curve is monotonically increasing between $(0,0)$ and $(1,1)$. Improved outlier detection performance leads to larger TPR and lower FPR values. Correspondingly, the ROC curve would be pushed towards the top left corner as shown in Fig. 5a, with the best possible AUROC value of 1.0. For example, there is only one customer failure in the industrial automotive microcontroller datasets we use. Correspondingly, there is a sharp transition in the ROC as shown in Fig. 5b, where the green area ($1 - AUROC$ where only one failure exists) can be interpreted as the yield loss when no defect escape occurs.

Compared to other metrics like Area Under Precision-Recall (AUPR), AUROC takes a more balanced consideration over both abnormal and normal data, which efficiently captures outliers and minimizes yield loss.

C. Review of Traditional Unsupervised Learning Models

Here gives a brief review of 4 different unsupervised anomaly detection methods: Gaussian model, One-class SVM, Isolation forest and Autoencoder.

1) Gaussian Model: One typical category of anomaly detection methods is to estimate the data distribution given the training samples, and then mark low-probability instances as anomaly. Gaussian model [26] is one of the most popular assumptions for the data distribution, which especially suits for the post-silicon test, as most of data follows a Gaussian distribution. Specifically, given N normal training samples $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\}$, the mean and covariance matrix are estimated as follows.

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}^{(i)} \tag{2}$$

$$\hat{\mathbf{C}} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}^{(i)} - \hat{\mu}\right)\left(\mathbf{x}^{(i)} - \hat{\mu}\right)^{\mathrm{T}} \tag{3}$$

Thus, clearly, the score function is defined as the probability distribution of the estimated Gaussian distribution as below.

$$s\left(\mathbf{x}\right) = \mathcal{N}\left(\mathbf{x}; \hat{\mu}, \hat{\mathbf{C}}\right) \tag{4}$$

2) One-class SVM: One-class support vector machine (OCSVM) proposed in [18] separates all the data from the origin with maximum margin in the feature space corresponding to the kernel function. In general, instead of directly looking at the probability distribution of the normal sample occurrence, one-class SVM attempts to map the data into a feature space and enclose the normal data into a small region. This results in a binary function which captures regions in the input space where most of the normal data live. The training process of one-class SVM is governed by a quadratic programming minimization problem as stated below.

$$\min_{\omega,\xi_i,\rho} \ \frac{1}{2}\left\|\omega\right\|^2 + \frac{1}{\nu N}\sum_{i=1}^{N}\xi_i - \rho; \tag{5}$$

$$\text{s.t. } \omega \cdot \phi\left(\mathbf{x}^{(i)}\right) \geq \rho - \xi_i; \tag{6}$$

$$\xi_i \geq 0, \tag{7}$$

where $\phi\left(\cdot\right)$ specifies the kernel function to be used for which radial basis function (RBF) is a common choice, ν is a hyperparameter characterizing the solution by setting the upper bound of the outliers inside the training dataset and the lower bound for the number of support vectors. According to the distance to the decision boundary in the feature space, represented by ρ, the score function can be depicted using

$$s\left(\mathbf{x}\right) = \omega \cdot \phi\left(\mathbf{x}^{(i)}\right) - \rho \tag{8}$$

Note that the signed version of the previous function is used to give a binary classification of the anomaly detection in [18].

3) Isolation forest: In addition to the distribution estimation and data separation in the feature space, a tree-based method called isolation forest [17] takes a disparate approach to distinguish the anomaly from the normal data. The isolation forest consists of an ensemble of isolation trees (iTrees). More formally, for each node T in an iTree, T is either an external node with no child or an internal node with one test and exactly two children (T_L, T_R). The test at node T consists of an attribute q and a split value p, and based on whether $q < p$ it will traverse the data point to either T_L or T_R. In order to build such an iTree, a subset of entire dataset $\mathbf{X}' \subset \{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\}$ is randomly selected. The iTree

is generated by recursively partitioning, and then a training algorithm recursively partitions $\mathbf{X}'$ by randomly selecting an attribute q and a split value p until there is only one data point in the node or all the data share the same value.

The underlying principle of anomaly detection for isolation forest is that the anomaly data is more likely to reach an external node with a smaller height (distance to the iTree root), as all the iTrees are generated randomly. Therefore, its score function is characterized by the average height to reach the external nodes in the ensemble of iTrees as follow.

$$s\left(\mathbf{x}\right) = -2^{-\mathbf{E}h(\mathbf{x})/c(N)}, \tag{9}$$

where $c\left(N\right)$ is a constant normalization factor related to the sample size N. The negative sign is added to be consistent with the definition of score function in this paper.

4) Autoencoder: There exists another popular anomaly detection method which is reconstruction-based enabled by an autoencoder. Similar ideas using reconstruction for anomaly detection are widely used in recent work [16], [21], [22]. First, the original data $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^D$ is encoded (compressed) into a latent variable $\mathbf{z} = F_E\left(\mathbf{x}\right) \in \mathcal{Z} \subseteq \mathbb{R}^d$ with $d \ll D$ typically, and then decoded (reconstructed) into the original space with $\mathbf{x}' = F_D\left(\mathbf{z}\right)$, as illustrated in Fig. 6. The encoder and the decoder are usually fully-connected neural networks for numerical data processing or convolutional neural networks for image processing. The entire autoencoder is trained to minimize the overall reconstruction error (loss function for training as shown below) so that a good embedding representation of the normal data can be learnt in the low-dimensional latent space $\mathcal{Z}$.

$$\mathcal{L} = \sum_{i=1}^{N} \left\| \mathbf{x}^{(i)} - \mathbf{x}'^{(i)} \right\|^2 \tag{10}$$

With an outlier data, since it doesn't follow the normal data embedding representation, it is expected to observe a large reconstruction error, which is used to defined the score function for the autoencoder.

$$s\left(\mathbf{x}\right) = -\left\|\mathbf{x} - F_D\left(F_E\left(\mathbf{x}\right)\right)\right\|^2 \tag{11}$$

III. Self-Labeling Unsupervised Outlier Detection

A. Self-Labeling via Transformation

Based on the discussions in the Section II, we aim to learn a robust and reliable score function $s : \mathcal{X} \to \mathbb{R}$ using normal training data only. Unsupervised learning for extremely-rare failure detection is challenging. Motivated by the self-labeling approach for anomaly image detection of [25], we convert this unsupervised learning problem to one that is based on multi-class classification with self-labeled training data.

Consider K distinct transformation functions $T^{(1)}, T^{(2)}, \cdots, T^{(K)}$, each defining a mapping to a m-dimensional feature space: $T^{(i)} : \mathbb{R}^n \to \mathbb{R}^m$, where $i \in [1, K]$. For a given training dataset X with N examples, we apply all K transformations to each sample, resulting in a transformed training dataset with KN examples. Each newly transformed example is labeled by its corresponding transformation applied. Formally, the resulting labeled training dataset is

$$\left\{\left(T^{(i)}\left(\mathbf{x}^{[j]}\right), T^{(i)}\right) | i \in [1, K], j \in [1, N]\right\} \tag{12}$$

Note that each label is given by the type of the transformation performed without involving any actual labeling effort nor abnormal data. The adopted transformations can be regarded as an approach for nonlinear feature extraction through which the original input space $\mathcal{X} \subseteq \mathbb{R}^n$ is mapped into a feature space $\mathcal{F} \subseteq \mathbb{R}^m$.

B. Proposed Self-labeling Unsupervised Outlier Detection

Fig. 7 highlights the proposed self-labeling outlier detection approach. During training, self-labeling is executed first per (12) to generate the transformed training dataset over which a multi-class classifier is trained to well classify each example to the corresponding label (transformation). During inference, similarly, given an input $\mathbf{x}$, K transformed inputs $\left\{T^{(1)}\left(\mathbf{x}\right), T^{(2)}\left(\mathbf{x}\right), \cdots, T^{(K)}\left(\mathbf{x}\right)\right\}$ are obtained by applying the K transformation functions. For each transformed $T^{(i)}\left(\mathbf{x}\right)$, the classifier outputs a K-dimensional probability vector $\mathbf{p}\left(T^{(i)}\left(\mathbf{x}\right)\right)$ with each k-th element specifying the predicted likelihood for $T^{(i)}\left(\mathbf{x}\right)$ to fall under class k, i.e. transformed by k-th transformation.

When the classifier is well trained using the normal data, it would classify a new unseen normal input $\mathbf{x}$ by outputting: $\mathbf{p}_i\left(T^{(i)}\left(\mathbf{x}\right)\right) \approx 1$ and $\mathbf{p}_i\left(T^{(k)}\left(\mathbf{x}\right)\right) \approx 0, i \neq k$ as the transformed unseen normal data is likely to locate within the transformed normal training data distribution. However, for an abnormal input, it is likely that $\mathbf{p}_i\left(T^{(i)}\left(\mathbf{x}\right)\right)$ is significantly lower than 1.0 as the transform outlier data may deviate from the transformed normal data distribution. Accordingly, we select the following score function

$$s\left(\mathbf{x}\right) = \sum_{i=1}^{K} \mathbf{p}_i\left(T^{(i)}\left(\mathbf{x}\right)\right) \tag{13}$$

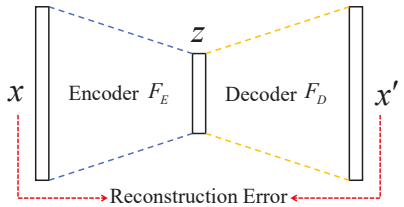


Fig. 6: Autoencoder illustration.

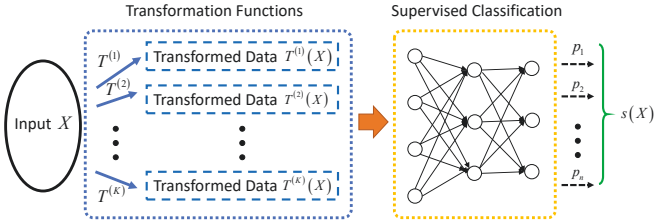


Fig. 7: Self-labeling unsupervised outlier detection framework.

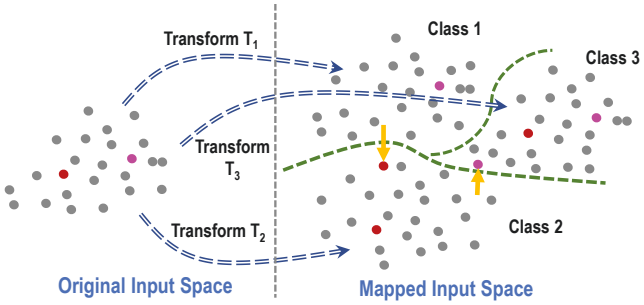


Fig. 8: Outlier detection via 3 transformations. Gray points: normal data; Red/pink points: outliers. Yellow arrows: misclassification of mapped outliers signifies anomaly detection.

An input is detected as an outlier if $s(\mathbf{x}) \ll K$. This self-labeling outlier detection is illustrated in Fig. 8.

IV. Design of Transformation Functions

One major challenge in the proposed self-labeling approach is to select proper transformation functions. For example, if $T^{(i)} = T^{(j)}$, the classifier cannot distinguish between the i-th and j-th class as they correspond to identical transformed input data location. Our key idea is to select a set of distinct information lossless transformation functions to retain sufficient statistics for the original data. Consider that our original data $\mathbf{x}$, the transformed data $T(\mathbf{x})$, and the final score function $s(\mathbf{x})$ formulate a Markov chain $s(\mathbf{x}) \longleftrightarrow T(\mathbf{x}) \longleftrightarrow \mathbf{x}$. According to the data processing inequality, we have the mutual information follows $I(s(\mathbf{x}); T(\mathbf{x})) \leq I(s(\mathbf{x}); \mathbf{x})$. In order to maximize the mutual information after the transformation, our method is to make the transformed data fully recoverable, without information loss, after the transformation.

Inspired by the recent developments in reversible neural networks [27], [28], we propose an reversible architecture for the transformation functions to fully retain the original data information. Suppose $\mathbf{x} = (x_1, x_2, \cdots, x_D)^{\mathrm{T}}$ has D features, we partition the indices of the D features into two sets p_1 and p_2 with equal size (add one artificial feature if D is odd), where $p_1 \cap p_2 = \emptyset$ and $p_1 \cup p_2 = [1, D] \cap \mathbb{N}$. With that, we can obtain two new vectors with the corresponding features as $\mathbf{x}_{p1}$ and $\mathbf{x}_{p2}$. Then the transformed data $\mathbf{y}$ (combining two parts $\mathbf{y}_{p1}$ and $\mathbf{y}_{p2}$) is given by the reversible transformation as follows.

$$\mathbf{y}_{p1} = \mathbf{x}_{p1} + G(\mathbf{y}_{p2}) \quad (14)$$

$$\mathbf{y}_{p2} = \mathbf{x}_{p2} + F(\mathbf{x}_{p1}) \quad (15)$$

where F and G are two arbitrary functions. Through solving the previous equations, the original input $\mathbf{x}$ can be fully recovered with knowing the output $\mathbf{y}$, showing the information lossless property of the reversible transformation block. In order to easily generate a large number of distinct transformations, the transformation function should be flexible enough to be configured. We considered three different approaches to increase the flexibility of the transformation: 1) function choices for the reversible block; 2) feature permutation; 3) cascade architecture.

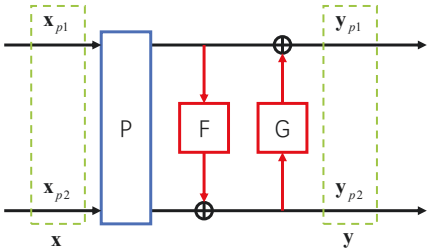


Fig. 9: Reversible lossless transformation block.

A. Function Choices for the Reversible Block

The arbitrariness of the two functions F and G provides a lot of freedom to be designed. Without adding additional training and computational cost for F and G, which are usually two neural networks, from our empirical experience, we suggest to apply two simple univariate nonlinear functions f and g for each element of the input vector. Therefore, we have $F(\mathbf{x}) = \left(f(x_1; \theta_f), f(x_2; \theta_f), \cdots, f\left(x_{D/2}; \theta_f\right)\right)^{\mathrm{T}}$ and $G(\mathbf{x}) = \left(g(x_1; \theta_g), g(x_2; \theta_g), \cdots, g\left(x_{D/2}; \theta_g\right)\right)^{\mathrm{T}}$, where $\theta_f \in \Theta$ and $\theta_g \in \Theta$ are the parameters for the two univariate functions from a parameter space Θ. Hence, we can create a pool of function choices, for each reversible block, we can randomly assign two functions to f and g and sample the parameters θ_f and θ_g from Θ to generate a large number of distinct transformations before training phase.

In particular, for our experimental settings, a pool of three different polynomial functions with randomized order parameter θ uniformly chosen from $\Theta = [2, \theta_{max}] \cap \mathbb{N}$ is applied, including: power polynomial functions,

$$p^{(\theta)}(x) = x^{\theta} \quad (16)$$

Legendre polynomial functions,

$$l^{(0)}(x) = 1 \quad (17)$$

$$l^{(1)}(x) = x \quad (18)$$

$$l^{(\theta)}(x) = \frac{2\theta - 1}{\theta} x l^{(\theta-1)}(x) - \frac{\theta - 1}{\theta} l^{(\theta-2)}(x) \quad (19)$$

and Chebyshev polynomial functions,

$$c^{(0)}(x) = 1 \quad (20)$$

$$c^{(1)}(x) = x \quad (21)$$

$$c^{(\theta)}(x) = 2x c^{(\theta-1)}(x) - c^{(\theta-2)}(x). \quad (22)$$

B. Feature Permutation

Beyond the function choice for each reversible block, we also add one more permutation block, as shown in Fig. 9, to boost the feature mixing and increase flexibility for distinct function generation. For the input of each reversible block, a random feature permutation (partition) is generated before training process and then fixed during the training and inference phase to produce two groups of data.

This technique boosts the diversity among different reversible blocks. Furthermore, with the help of the cascade architecture (introduced in the next section), this permutation will also boost feature mixing among multiple features, which

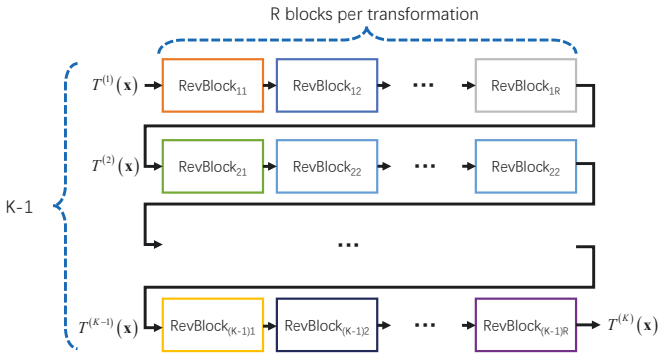


Fig. 10: Reversible transformation architecture.

TABLE I: Public anomaly detection datasets.

Public	# Samples	# Features	# Anomaly
thyroid	3772	6	93 (2.5%)
glass	214	9	9 (4.2%)
Satimage-2	5803	36	71 (1.2%)
shuttle	49097	9	3511 (7%)
smtp	95156	3	30 (0.03%)
speech	3686	400	61 (1.65%)

serves a nonlinear feature extraction process providing a distinct view of the normal data for each transformation.

C. Overall Cascade Reversible Architecture

The overall reversible lossless transformation architecture is illustrated in Fig. 10. Each reversible block is marked in different color to indicate they are all distinct. We apply both techniques: function choices and feature permutation, described in Section IV-A and IV-B, to generate a large number of distinct reversible blocks.

Here, a cascade architecture is applied. Note that the reversible lossless tranformation block can be repeated multiple times to get a single tranformation. Hence, the data will go through R reversible blocks to acquire $T^{(i+1)}(\mathbf{x})$ from $T^{(i)}(\mathbf{x})$. In order to generate K different transformations, this computation is performed $K-1$ times, assuming the first transformation is the original data point $T^{(1)}(\mathbf{x}) = \mathbf{x}$, which is already available. Thus, there exists $(K-1)R$ different reversible blocks in the entire architecture, providing a large number of distinct transformations without losing any information.

V. Experimental Results

A. Methods and Datasets

Using six public outlier detection datasets [29], [30] and post-silicon testing datasets for six real customer failures/returns of an advanced industrial automotive microcontroller, We demonstrate and compare the performances of several unsupervised learning methods: Gaussian model [26], one-class SVM [18], isolation forest [17], autoencoder [16], and four configurations of the proposed self-labeling outlier detection method.

For the industrial cases, the six customer failures were from several millions of parts shipped, showcasing the real-life challenges in extremely-rare defect detection. We pulled out the post-silicon testing data for the wafer lot containing the customer failure to be learned. There were around tens of thousands of parts in a wafer lot which were all manufactured around the same time with similar tools as the customer failure chip, thus providing relevant data for statistical outlier analysis. Parts in a wafer lot went through wafer test and final test at different temperatures, which we call test inserts. Each dataset (with one customer failure chip) consists of five to six test inserts. In total, we had 32 test inserts corresponding to the 6 datasets for the 6 customer failure chips. Each test insert contains up to a few thousands of parametric tests for tens of thousands of parts from the wafer lot which contains the single customer failure part. In addition, we also generate 6 additional datasets using the test inserts containing only the "critical" parametric tests (features) suggested by expert engineers empirically based on the customer failure modes, which we call "critical" tests (inserts). Hence, the number of parametric tests (features) varied from 6 to 5,592 in the test inserts. The detailed numbers of features and examples of the public and industrial datasets are listed in Table I and Table II, respectively. Furthermore, the samples with missing or invalid test values are discarded here. Therefore, For each test insert, the number of samples in the "critical" tests are larger than or equal to the one for the entire tests.

TABLE II: Industrial automotive microcontroller datasets.

Datasets		Entire tests		"Critical" tests	
Chip	Insert	# Samples	# Features	# Samples	# Features
Chip 1	A	47006	104	50101	6
	B	45617	1134	45658	103
	C	41828	989	42694	28
	D	38940	769	42396	28
	E	42293	1054	42293	31
Chip 2	A	44579	104	50498	6
	B	46529	1135	46563	103
	C	44833	989	45472	28
	D	42905	780	44843	28
	E	44586	1051	44586	31
Chip 3	A	38897	102	39840	6
	B	37826	695	37831	327
	C	34544	369	34544	181
	D	34245	352	34246	153
	E	34228	1377	34228	690
Chip 4	A	16115	184	17356	6
	B	14892	379	15238	24
	C	13581	2558	13692	128
	D	13169	2587	13169	58
	E	13272	2396	13272	47
	F	7889	5592	7888	59
Chip 5	A	44282	105	50029	6
	B	47257	1151	47350	117
	C	43007	982	44862	32
	D	43893	800	44522	32
	E	44036	1108	44036	37
Chip 6	A	16591	241	18403	6
	B	16514	1521	16530	255
	C	15268	3009	16157	129
	D	15916	2645	16081	60
	E	16068	4224	16068	47
	F	16054	5526	16054	63

Fig. 11 provides the visualization of the data distribution of one example test insert (Chip 6 with the "critical" test insert A), with the single customer failure marked in orange. As we

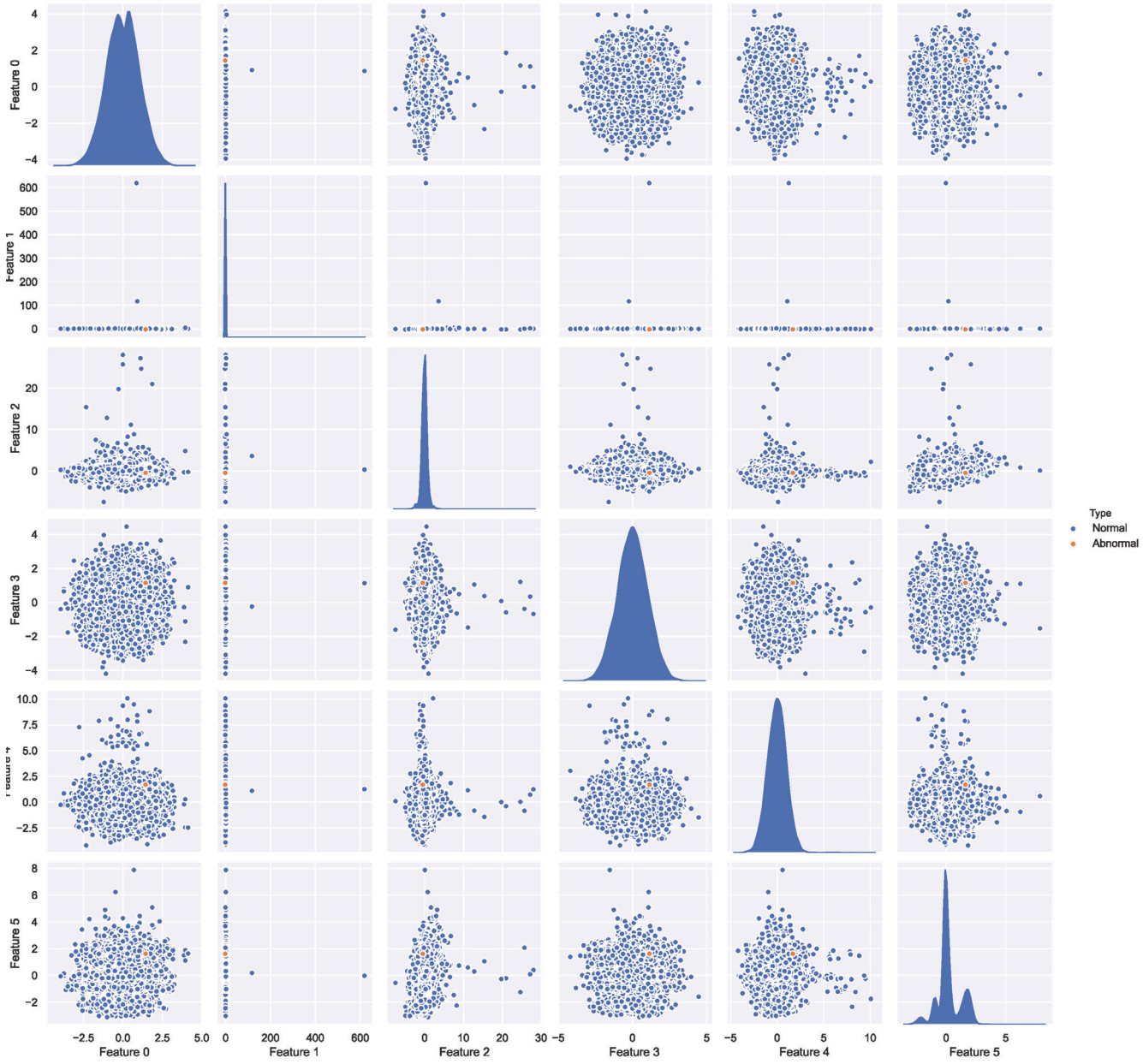


Fig. 11: Dataset example visualization.

can see, the outlier point almost locates at the center of the data distribution in every dimension, which makes it extremely challenging to separate the anomaly from the normal data by only checking each single parameter or test feature.

B. Experimented method settings

For each test insert, we trained a machine learning model based on each method to screen out the customer failure part. 90% of the normal data were randomly sampled using a uniform distribution as the training dataset, and remaining 10% of normal data and the single customer failure were used as testing data.

We compared our proposed self-labeling method with several popular families of existing outlier detection techniques as reviewed in Section II. Under the category of input-distribution-based approaches, we applied a Gaussian distribution model [26] to capture the mean vector and covariance matrix of normal data, and employed the probability density function as the score function. Furthermore, we adopted one-class support vector machine (OCSVM) [18] which utilized the radial basis function (RBF) kernel and its decision function as its score function. The isolation forest method [17] generated a forest of 250 trees with a resampling size of 1024, the average path length metric suggested in [17] was regarded as the score function. Moreover, we also investigated the popular reconstruction-based method, autoencoder [16]. An autoencoder with a latent size of 32 and a hidden layer of size 64 was trained over 100 epochs with a batch size of 64,

and the reconstruction error of the autoencoder was selected as its corresponding score function.

For the proposed self-labeling outlier detection method, we experimented four variants of the proposed self-labeling outlier detection method. This first one, marked as *PolyTrans* later, directly used the polynomial functions as the transformation functions, and the other three adopted the reversible lossless transformation blocks. As mentioned in Section IV, the function pool for reversible block consists of three families of univariate polynomial functions including polynomial functions, Legendre functions and Chebyshev functions. The detailed configurations for the three reversible transformations are listed: 1) $\theta_{max} = 4$ and $R = 1$ for *RevTransA*; 2) $\theta_{max} = 5$ and $R = 1$ for *RevTransB*; 3) $\theta_{max} = 5$ and $R = 2$ for *RevTransC*. In order to obtain a fair number of distinct transformations, all the experimented variants used $K = 22$ transformations. The classifier used was an artificial neural network with two hidden layers of size 64 and 16 trained over 20 epochs using a batch size of 64. The same settings were used for all the public datasets and all the real industrial automotive microcontroller datasets. Furthermore, we applied L2 regularization with a weight decay factor of 0.0005 for the classifier training to avoid overfitting and improve the overall performance.

C. Public Dataset Performance

Table III provided a detailed comparison of different methods for all the public datasets. As discussed in Section II-B, we use AUROC as a measure of performance for fair comparison between different methods. Although the public datasets in I don't have extremely rare outliers compared to the industrial post-silicon testing datasets, they still serve as good basis for general outlier detection method comparison. Among all the 8 methods considered, *RevTransC* gives the best AUROC results on average. Although the other three variants of the proposed methods didn't outperform isolation forest in terms of average AUROC, they are still comparable (within 1%) to it, which is the best reference method, and surpass other three reference methods a lot, demonstrating that the proposed methods can be applied as a robust general-purpose outlier detection method as well.

D. Industrial Dataset Performance

The performance comparison for the the industrial automotive microcontroller datasets are reported in Table IV, V and VI. Each industrial dataset (chip) contains multiple test inserts with a single customer failure, and we learnt a machine learning model for each test insert to screen out the particular customer failure based on each method or setting.

During the advanced outlier detection phase performed at the very end of the testing process, as shown in Fig. 1, experienced engineers may determine whether it is worthwhile to discard a certain number of chips, i.e. at a given yield loss level, in order to screen out the customer failure based on outlier detection results collected from all test inserts. As long as the outlier detection results from one of the test inserts can screen out the customer failure, the outlier detection performance based on other test inserts doesn't matter. Therefore, we report the best performance value among all the test inserts for each method in these three tables.

Beyond AUROC introduced in Section II-B, we also used another performance metric, *estimated yield*, to evaluate different methods for the industrial datasets in Table V and VI. We define the estimated yield of each outlier detection method as one minus the percentage of normal chips we will reject together with the customer failure from the corresponding wafer lot. Recall that the proposed machine learning based advanced outlier detection takes place after the preceding wafer/package/final test steps (Fig. 1) in which each test insert will reject a certain number of chips based on functional tests and hard limits in parametric tests. The normal chips that we will reject together with the customer failure by performing the outlier detection at a specific test insert (e.g. wafer test) might have already failed in a subsequent test insert (e.g. final test). Hence, the estimated yield of each outlier detect method is calculated excluding those chips which already failed in subsequent test inserts. As stated before, the final decision of advanced outlier detection process is largely based on engineering experience. The estimated yield, in some sense, reflects the minimum over-reject we can expect in order to remove the particular customer failure in the advanced outlier detection phase.

Note that the adopted industrial datasets only have a single customer failure for each dataset. For each dataset, the reported AUROC and estimated yield value varies for different methods and different datasets, manifesting the real life challenges in extremely-rare defect detection. On average, the reference Gaussian method produces the worst AUROC and estimated yield performance, which mispredicts examples close to the center of the normal data distribution to be abnormal. This suggests that directly characterizing normal data using a Gaussian distribution is not effective in separating out hard latent defects. As for the other reference methods, isolation forest and autoencoder tend to give a relatively good performance among all the reference methods, as shown in Table IV to VI.

In general, the four variants of the proposed self-labeling outlier detection method are among the very best of all the experimented methods. Specifically, the variant *RevTransC* reports best performance values in Table IV and V, and the variant *PolyTrans* reports the best estimated yield in Table VI. Under this extremely-rare outlier detection context, the proposed self-labeling approach is able to expose the uniqueness of the hard-to-detect outliers and maximize the chance of detecting such extremely-rare (latent) defects (e.g. the targeted customer failure part) as well as minimize the overkill (false positive) rate.

In practice, for cost reason, test engineers may set a minimum yield goal for an outlier detection method to screen out potential customer failures or returns. In Table V and VI, the number of customer failures out of the total six that can be screened out under a yield goal of 93% by each method is reported under the row "# Y$\geq$93%". In other words, "#

TABLE III: Comparison of outlier detection performance using AUROC for public datasets.

Datasets	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
thyroid	0.974	0.937	0.989	0.946	0.955	0.962	0.971	0.94
glass	0.683	0.513	0.794	0.624	0.899	0.825	0.767	0.815
satimage-2	0.995	0.981	0.996	0.93	0.995	0.991	0.992	0.985
shuttle	0.994	0.983	0.998	0.998	1	0.995	0.992	0.999
smtp	0.815	0.773	0.921	0.859	0.729	0.86	0.866	0.963
speech	0.515	0.461	0.45	0.482	0.542	0.508	0.505	0.525
Avg. AUROC	0.829	0.775	0.858	0.807	0.853	0.857	0.849	**0.871**

TABLE IV: Comparison of outlier detection performance using AUROC for industrial datasets.

Datasets	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
Chip 1	0.685	0.872	0.882	0.902	0.461	0.889	0.938	0.949
Chip 2	0.823	0.964	0.984	0.979	0.898	0.928	0.939	0.895
Chip 3	0.882	0.94	0.937	0.977	0.928	0.934	0.954	0.973
Chip 4	0.921	0.788	0.838	0.931	0.826	0.828	0.814	0.843
Chip 5	0.972	0.907	0.978	0.959	0.986	0.865	0.966	0.985
Chip 6	0.929	0.738	0.758	0.888	0.916	0.891	0.967	0.999
Chip 1 (“Critical”)	0.858	0.753	0.885	0.759	0.839	0.818	0.922	0.647
Chip 2 (“Critical”)	0.846	0.858	0.853	0.845	0.917	0.977	0.952	0.925
Chip 3 (“Critical”)	0.587	0.874	0.932	0.733	0.992	0.815	0.826	0.986
Chip 4 (“Critical”)	0.81	0.77	0.586	0.944	0.778	0.809	0.529	0.762
Chip 5 (“Critical”)	0.841	0.99	0.984	0.671	0.942	0.96	0.936	0.941
Chip 6 (“Critical”)	0.89	0.861	0.963	0.818	0.935	0.929	0.899	0.896
Avg. AUROC	0.837	0.860	0.882	0.867	0.868	0.887	0.887	**0.900**

TABLE V: Comparison of outlier detection performance using estimated yield for industrial datasets containing the entire parametric tests.

Datasets	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
Chip 1	0.779	0.917	0.904	0.921	0.543	0.907	0.948	0.961
Chip 2	0.853	0.971	0.987	0.982	0.914	0.944	0.953	0.919
Chip 3	0.886	0.941	0.939	0.978	0.943	0.937	0.96	0.974
Chip 4	0.978	0.914	0.912	0.978	0.932	0.91	0.915	0.933
Chip 5	0.984	0.934	0.99	0.971	0.989	0.894	0.974	0.99
Chip 6	0.947	0.792	0.787	0.916	0.931	0.904	0.974	0.999
Avg. Yield	0.905	0.912	0.920	0.958	0.875	0.916	0.954	**0.963**
# Y≥93%	3	3	3	4	4	2	**5**	**5**

TABLE VI: Comparison of outlier detection performance using estimated yield for industrial datasets containing the “critical” parametric tests.

Datasets (“Critical”)	Reference Methods				Proposed Methods			
	Gaussian	OCSVM	IsoForest	AE	PolyTrans	RevTransA	RevTransB	RevTransC
Chip 1	0.902	0.795	0.921	0.797	0.893	0.865	0.934	0.728
Chip 2	0.897	0.899	0.881	0.891	0.933	0.982	0.969	0.95
Chip 3	0.632	0.878	0.934	0.743	0.993	0.821	0.832	0.989
Chip 4	0.867	0.922	0.858	0.965	0.888	0.897	0.811	0.915
Chip 5	0.872	0.994	0.99	0.76	0.966	0.965	0.951	0.953
Chip 6	0.919	0.897	0.976	0.852	0.958	0.934	0.925	0.914
Avg. Yield	0.848	0.898	0.927	0.835	**0.939**	0.911	0.904	0.908
# Y≥93%	0	1	3	1	**4**	3	3	3

Y≥93%” counts all the chips with the estimated yield no less than the yield goal (93%) for each method. Given the fact that the customer failures under this study escaped from the original production testing on several millions of parts shipped, robustly screening out even one or two of these customer failures is already challenging. Among all the methods, our proposed self-labeling outlier detection methods screen out the most number of customer failures under the given yield goal. In particular, in Table V *RevTransB* and *RevTransC* can catch five customer failures out of the total six, and in Table VI *PolyTrans* can catch four customer failures while meeting the minimum 93% yield goal.

As we can observe from Table V and VI, the overall estimated yield and the number of customer failures captured under the given yield goal reduce as we only consider the “critical” parametric tests. This may suggest that manual selection of critical test features based on empirical experiences can lead to sub-optimal outcomes and exploiting powerful machine learning techniques capable of considering a large number of features may be advantageous. Even with less information (parametric tests) about the failure part, the proposed methods still retain a high estimated yield and surpass most of the reference methods as shown in Table VI. Specifically, *PolyTrans* gives the best average estimated yield of 93.9% among all the experimented methods.

We compare the four variants of the proposed method based on the results from Table III to V. The three variants of self-labeling method based on reversible transformations

achieve a better average AUROC and estimated yield on average compared to *PolyTrans*, suggesting that maintaining the raw feature information is critical to achieve good outlier detection performance when designing the transformations for the self-labeling method. Furthermore, *RevTransC* achieves the best performance among all three reversible transformation variants, demonstrating that more distinct transformations with higher order and more complex transformations tend to work better. There exists one exception. Table VI reports that *PolyTrans* gives the best yield among all methods while not all available parametric tests are considered. We expect that the the test inserts included in those datasets may not contain all information relevant to the particular customer failure, making the reversible transformations less effective.

VI. Conclusion

We presented a machine learning enabled outlier detection methodology in order to facilitate the screening of extremely-rare failures that have escaped from the standard post-silicon testing flow. We proposed a self-labeling technique for unsupervised outlier detection through transformations of available test data, effectively exposing the abnormal behaviour of extremely-rare chip failures in a high-dimensional test feature space. Based on public-domain outlier detection and challenging industrial automotive microcontroller test datasets, we demonstrated through extensive experimental studies that our proposed method consistently outperforms other popular outlier detection algorithms in terms of accuracy and robustness, leading to reduction of yield loss and test escape.

Acknowledgement

This material is based upon work supported by the Semiconductor Research Corporation (SRC) through Texas Analog Center of Excellence at the University of Texas at Dallas (Task ID: 2810.031) and by the National Science Foundation under Grants No. 1956313.

The authors would like to thank NXP Semiconductors for providing the experimental data, and University of California Santa Barbara (UCSB) Center for Scientific Computing (CSC) at California NanoSystems Institute (CNSI) for providing computing support. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of SRC, NSF, NXP Semiconductors, University of California, Santa Barbara, and their contractors.

References

[1] W. R. Mann, F. L. Taber, P. W. Seitzer, and J. J. Broz, "The leading edge of production wafer probe test technology," in *Proceedings of International Conference on Test*, Oct 2004, pp. 1168–1195.

[2] S. S. Sabade and D. M. Walker, "Ic outlier identification using multiple test metrics," *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 586–595, Nov 2005.

[3] S. Illyes and D. A. G. Baglee, "Statistical bin limits-an approach to wafer disposition in ic fabrication," *IEEE Transactions on Semiconductor Manufacturing*, vol. 5, no. 1, pp. 59–61, Feb 1992.

[4] M. J. Moreno Lizaranzu, "Computer integrated manufacturing in semiconductor industry. automation, electronic wafer mapping, defect reduction and equipment utilization improvement in probe and final test," Ph.D. dissertation, University of Seville, November 2012.

[5] "Guidelines for part average testing," *Automotive Electronic Council*, vol. AEC-Q001 Rev-D, Dec. 2011.

[6] T. Sakamoto, K. Yofu, and T. Kyuho, "New method of screening out outlier; expanded part average testing during package level test," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 4, pp. 351–356, Nov 2017.

[7] W. Dobbelaere, R. Vanhooren, W. De Man, K. Matthijs, A. Coyette, B. Esen, and G. Gielen, "Analog fault coverage improvement using final-test dynamic part average testing," in *Proceedings of IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–9.

[8] M. J. Moreno-Lizaranzu and F. Cuesta, "Improving electronic sensor reliability by robust outlier screening," *Sensors (Basel, Switzerland)*, vol. 13, no. 10, pp. 13 521–13 542, 2013.

[9] K. M. Butler, S. Subramaniam, A. Nahar, J. M. Carulli, T. J. Anderson, and W. R. Daasch, "Successful development and implementation of statistical outlier techniques on 90nm and 65nm process driver devices," in *IEEE Intl. Reliability Physics Symposium*, 2006, pp. 552–559.

[10] W. R. Daasch, J. McNames, D. Bockelman, and K. Cota, "Variance reduction using wafer patterns in i/sub ddq/ data," in *International Test Conference*, 2000, pp. 189–198.

[11] S. S. Sabade and D. M. H. Walker, "Comparison of wafer-level spatial i/sub ddq/ estimation methods: Nnr versus ncr," in *IEEE Intl. Workshop on Current and Defect Based Testing*, Apr. 2004, pp. 17–22.

[12] W. R. Daasch, K. Cota, J. McNames, and R. Madge, "Neighbor selection for variance reduction in iddq and other parametric data," in *IEEE International Test Conference*, USA, 2001, pp. 92–100.

[13] P. M. O'Neill, "Production multivariate outlier detection using principal components," in *IEEE International Test Conference*, 2008, pp. 1–10.

[14] N. Sumikawa, J. Tikkanen, L. Wang, L. Winemberg, and M. S. Abadir, "Screening customer returns with multivariate test analysis," in *IEEE International Test Conference*, 2012, pp. 1–10.

[15] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009.

[16] R. Chalapathy and S. Chawla, "Deep Learning for Anomaly Detection: A Survey," *arXiv:1901.03407*, Jan. 2019.

[17] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation forest," in *Proceedings of IEEE International Conference on Data Mining*, Dec 2008, pp. 413–422.

[18] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[19] D. M. Tax and R. P. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, Jan 2004.

[20] L. Ruff *et al.*, "Deep One-Class Classification," in *International Conference on Machine Learning*, Jul. 2018, pp. 4393–4402.

[21] B. Zong *et al.*, "Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection," in *International Conference on Learning Representations*, Feb. 2018.

[22] S. Zhai, Y. Cheng, W. Lu, and Z. Zhang, "Deep Structured Energy Based Models for Anomaly Detection," in *Proceedings of International Conference on Machine Learning*, Jun. 2016, pp. 1100–1109.

[23] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, "A Survey on GANs for Anomaly Detection," *arXiv:1906.11632*, Jun. 2019.

[24] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery," in *Information Processing in Medical Imaging*, Cham, 2017, vol. 10265, pp. 146–157.

[25] I. Golan and R. El-Yaniv, "Deep Anomaly Detection Using Geometric Transformations," in *Proceedings of Advances in Neural Information Processing Systems*, 2018, pp. 9758–9769.

[26] P. J. Rousseeuw and K. V. Driessen, "A fast algorithm for the minimum covariance determinant estimator," *Technometrics*, vol. 41, no. 3, pp. 212–223, 1999.

[27] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: backpropagation without storing activations," in *Advances in neural information processing systems*, 2017, pp. 2214–2224.

[28] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, "Reversible architectures for arbitrarily deep residual neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] S. Rayana, "ODDS library," 2016. [Online]. Available: http://odds.cs.stonybrook.edu

[30] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml