University of California
Santa Barbara

Learning from Production Test Data: From Statistical Characterization to Modeling for Anomaly Detection

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Electrical and Computer Engineering

by

Fan Lin

Committee in charge:

Professor Kwang-Ting Tim Cheng, Chair
Professor Malgorzata Marek-Sadowska
Professor Alberto Giovanni Busetto
Professor XiFeng Yan

June 2016

The Dissertation of Fan Lin is approved.

Professor Malgorzata Marek-Sadowska

Professor Alberto Giovanni Busetto

Professor XiFeng Yan

Professor Kwang-Ting Tim Cheng, Committee Chair

June 2016

Learning from Production Test Data: From Statistical Characterization to Modeling for Anomaly Detection

by

Fan Lin

Acknowledgements

I deeply appreciate the guidance of Prof. Tim Cheng through the years. I have been lucky to have learned and changed so much during the PhD study. I would also like to acknowledge my committee, Prof. Malgorzata Marek-Sadowska, Prof. Alberto Giovanni Busetto, and Prof. XiFeng Yan for their input. Thanks to Chun-Kai Hsu for the collaborative work on the project, and all my labmates and friends for simply being there. I appreciate the inspiring discussions with Chieh-Chi Kao, Kuo-Chin Lien, and the other members of our image processing study group. Finally, thanks for the support from my parents and my girl friend Stephanie.

Curriculum Vitæ

Fan Lin

Education

2016	Ph.D. in Electrical and Computer Engineering, University of California, Santa Barbara, United States.
2014	M.S. in Electrical and Computer Engineering, University of California, Santa Barbara, United States.
2010	B.S. in Electrical Engineering, National Taiwan University, Taipei, Taiwan.

Experience

2011 – 2016	Graduate Research Assistant, University of California, Santa Barbara, United States.
2015	Test Engineer Intern, Oracle, Santa Clara CA, United States.
2013	Test Engineer Intern, Broadcom, Irvine CA, United states.
2012	Test Engineer Inter, Taiwan Semiconductor Manufacturing Company (TSMC), Taiwan.
2012 – 2014	Teaching Assistant, University of California, Santa Barbara, United States.

Honors and Awards

2016	Dissertation Fellowship, University of California, Santa Barbara.

2015	Studying Abroad Scholarship, Ministry of Education, Taiwan.
2010	Research Award, Lam Research Corp., Taiwan.
2010	Valedictorian Speech, National Taiwan University, Taiwan.
2010	Presidential Award, National Taiwan University, Taiwan.
2010	Innovation Award, National Taiwan University, Taiwan.
2010	Special Project Award, National Taiwan University, Taiwan.

Publications

F. Lin and K.-T. Cheng, "An Artificial Neural Network Approach for Screening Test Escapes," submitted to *International Test Conference (ITC)*, 2016.

F. Lin, C.-K. Hsu, and K.-T. Cheng, "Pairwise Proximity-Based Features for Test Escape Screening," in *International Conference on Computer-Aided Design (ICCAD)*, 2015.

F. Lin, C.-K. Hsu, and K.-T. Cheng, "AdaTest: an Efficient Statistical Test Framework for Test Escape Screening," in *International Test Conference (ITC)*, 2015.

F. Lin, C.-K. Hsu, and K.-T. Cheng, "Learning from Production Test Data: Correlation Exploration and Feature Engineering," in *Asian Test Symposium (ATS)*, 2014.

F. Lin, C.-K. Hsu, and K.-T. Cheng, "Feature Engineering with Canonical Analysis for Effective Statistical Tests Screening Test Escapes," in *International Test Conference (ITC)*, 2014.

S. Zhang, **F. Lin**, C.-K. Hsu, K.-T. Cheng, and H. Wang, "Feature Engineering with Canonical Analysis for Effective Statistical Tests Screening Test Escapes," in *International Test Conference (ITC)*, 2014.

C.-K. Hsu, **F. Lin**, K.-T. Cheng, W. Zhang, X. Li, J. M. Carulli, and K. M. Butler, "Test Data Analytics: Exploring Spatial and Test-Item Correlations in Production Test Data," in *International Test Conference (ITC)*, 2013.

Abstract

Learning from Production Test Data: From Statistical Characterization to Modeling for Anomaly Detection

by

Fan Lin

Modern test programs for post-silicon testing include a large number of test measurements applied in multiple settings such as different temperatures, supply voltages, and operation modes to meet the demanding quality requirements of the products. In addition to the pass/fail results of each test item, there exist multiple types of correlations in the huge amount of production test data. Identifying and modeling the hidden correlations in the test data could help screen test escapes, which are chips that pass all test items but fail in system-level application.

This thesis focuses on developing revealing features and machine learning algorithms for classifying test escapes based on production test data. In terms of feature engineering, three types of feature sets that represent different aspects of how a chip deviates from the normal population are proposed. In addition, a linear transformation that compacts the critical information for feature reduction and a collection of nonlinear transformations that reveal additional abnormalities of the test escapes are proposed to effectively expose the test escapes as outliers in certain perspectives. We have also developed frameworks exploiting state-of-the-art machine learning algorithms including a support vector machine (SVM), a cascade of AdaBoost classifiers, and an artificial neural network.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the growing complexity and the shrinking size of modern chips, more tests have been added to the test program to assure the quality of a product, and tests are often applied multiple times under different environment settings to cover design corners. The increasing amount of tests has resulted in excessive test time and massive test data. With the help of effective data analytics, such test data has been transformed from a by-product of little value to a great source of information for better understanding the device under test (DUT). This dissertation explores and develops machine learning techniques for detecting test escapes, which are chips that pass the entire test program but fail at system-level applications, based on semiconductor production test data.

1.1 Correlations in Production Test Data

There exist meaningful correlations in the test data. In general, test data correlations can be classified into three types: *spatial correlations*, *inter-test-item*

correlations, and *temporal correlations*. Spatial patterns on a wafer have been commonly observed for various test measurements. The existence of a pattern implies that the measurement of a die is somehow correlated to the measurements of the other dies on the same wafer. The cause of spatial patterns could range from manufacturing processes, equipment settings, to test configurations such as a multi-site configuration.

There often exist correlations among measurements of multiple test items in a test program. Examples of strong inter-test-item correlations include the same test applied multiple times under different electrical and/or environmental settings and different tests testing the same functionality of the chip. Once inter-test-item correlations are identified, test items in the test program can be reordered for more efficient defect screening, some redundant test items can be removed for test time reduction, and multivariate models can be constructed for outlier detection.

Measurements of the same test item for dies in different wafers intuitively should exhibit similar patterns if the manufacturing process is stable. Such temporal correlations, across wafers and lots for measurements at the same die location, can be used to model the variation over time and manufacture/test equipment. Monitoring the temporal correlations often reveals the stability, integrity, and robustness of the manufacturing and test processes and thus is very useful for debugging.

1.2 Machine Learning for Detecting Test Escapes

Even with the ever growing number of tests and a test often being applied multiple times under different electrical/environmental conditions, there still exist test escapes, because system-level tests are usually not applied to all of the chips before shipment. Test escapes often have various root causes and therefore are hard to be detected with a single test. Additionally, the amount of test escapes in a mature manufacture process are typically within the range of hundreds or tens of Parts Per Million (PPM), so there are only a limited number of samples to be learned from for detecting future test escapes.

In a machine learning process, the sample data set is first split into three sets: a training set, a testing set, and an optional validation set. A training set is used for the machine learning to build a model based on, and therefore, must be statistically representative of the entire sample space, i.e. the statistical characteristics the training set possesses should accurately reflect the statistical characteristics of the entire sample set. If the training set possesses some unique characteristics that do not generalize to the rest of the samples, the learned model will not apply to the rest of the model accurately. After the learning phase, the learned model is applied to the testing set for evaluation of the performance such as prediction accuracy in a regression task or the true positive rate and the false positive rate in a classification task. In some algorithms, e.g. a support vector machine (SVM), multiple models are built based on a search of optimal values for some parameters, and a validation set is used for evaluating the models for selecting the best one.

There are, however, some challenges when applying the abovementioned ma-

chine learning framework in semiconductor production test data. First, process variation exists and could accumulate over time. That is, the training dataset created based on a certain collection of existing samples may not represent samples that are manufactured later because the manufacturing process has induced variation in the product characteristics over time. Therefore, a model learned based on the early training dataset could not be applied to the later query products. To address this problem, we need to carefully remove the wafer-to-wafer and lot-to-lot variations and have the machine learning algorithm learn only the characteristics of the chips that are immune from process variations. The developed machine learning framework must also constantly monitor how the learned model fit the newly manufactured data, and if there is a significant difference in the performance, e.g. a sudden drop of prediction accuracy, the newly manufactured samples should be checked if they are anomalies. If not, a new training set should be created to represent the new characteristics of the manufactured samples and a new model should be built based on the new training set.

When using machine learning algorithms to detect test escapes, another problem is that there are typically only a very small amount of test escapes in millions of chips. Additionally, test escapes could result from very different root causes. Therefore, it is difficult to find a universal test for detecting all test escapes. Instead, a general strategy in this dissertation is to explore multiple perspectives which could potentially reveal some abnormalities of the test escapes, and let the machine learning algorithm automatically extract the most critical information from the many perspectives to expose test escapes as anomalies.

1.3 Proposed Methods

It has been demonstrated that statistical analysis based on production test data, also known as *statistical tests* [1], could detect test escapes as anomalies. This dissertation further explores various state-of-the-art machine learning techniques and develops effective statistical tests.

In Chapter 2, we propose using a *residual vector*, which is the difference between the measured test values and some expected values, as the features for classification. Different expected values would result in different types of residual vectors, and potentially reveal different aspects of the chips under test. Therefore, our strategy is to include as many potentially useful residual vectors as possible. However, having too many features may not only increase the runtime but also deteriorate the accuracy for a machine learning classifier. To address this problem, a linear transformation called *canonical analysis* is proposed. Canonical analysis could compact the separation between classes of samples in a high-dimensional feature space into the first few dimensions in a transformed feature space, therefore it is used for feature reduction, followed by a classic SVM classification. The experimental results show that canonical analysis could significantly reduce the runtime of SVM and in some cases, improve the accuracy of SVM for classifying test escapes.

The framework incorporating canonical analysis and SVM, however, requires all potentially useful features (multiple types of residual vectors) to be generated first before canonical analysis can be applied during test application, which results in significant runtime and memory usage. Chapter 3 proposes using a popular framework in real-time face recognition called the *Viola-Jones* framework for

detecting test escapes. Named *AdaTest*, this framework is composed of a cascade of *adaptive boosting* (AdaBoost) classifiers. AdaBoost is an algorithm that combines multiple classifiers, referred to as the weak classifiers, into a final classifier, referred to as the strong classifier. In this framework, each weak classifier is a decision stump, which is simply a threshold set on a single feature. As a result, AdaTest would select only the most critical features without any transformation in the training phase for classification, and only these selected features need to be produced during test application. Therefore, AdaTest is significantly faster than the previous framework utilizing canonical analysis and SVM, and since there is no feature transformation in the process of learning a model, the selected features could be directly interpreted for the diagnosis of the test escapes.

In addition to canonical analysis, a collection of nonlinear transformations are introduced in Chapter 4 to reveal more abnormalities of test escapes. We calculate the pairwise proximity between each pair of samples on a wafer and include this information as the features for our analysis. To include as much potentially useful information as possible, we included seven proximity/distance functions as the nonlinear transformations that characterize the relation between of samples. For a wafer with N chips, each nonlinear transformation would result in one N by N symmetrical proximity matrix, whose elements represent the proximity of two corresponding samples. A technique called *constant shift embedding* is then applied to transform this proximity matrix back into a Euclidean vector representation so that traditional machine learning algorithms could be applied.

Chapter 5 introduces an artificial neural network (ANN) approach for detecting test escapes. Artificial neural networks have demonstrated great potential

recently in many applications such as image processing and voice recognition. For detecting test escapes, we design an *autoencoder* structure that is trained using only the good chip population in the training set, which is referred to as unsupervised learning. An autoencoder is an artificial neural network with the input and output layers both representing the original input features, and a bottleneck layer in which the number of neurons is smaller than that of the input/output layer. The bottleneck layer therefore contains information that is critical for representing the dataset. With the unsupervised learning process, the autoencoder fits only the good chip population and any query chip that the autoencoder model does not fit could be identified as an anomaly, which is likely a test escape.

Chapter 2

Canonical Analysis and SVM

2.1 Introduction

For many applications, the requirements of the defective parts per million (DPPM) of integrated circuits have to be extremely close to zero. Each field return found at the customer side incurs significant cost and requires thorough analysis of the cause. It has been shown that a good fraction of field returns are test escapes that pass the complete test program, but fail at system level due to their intrinsic defects [2, 3]. Applying system level tests to each chip prior to shipment, however, is undesired because it often results in high test time and cost.

This dissertation addresses the problem of identifying as many test escapes as possible by statistical analysis of the test data produced by a given test program, without taking any additional physical measurements. Such an approach can be viewed as adding statistical tests to the original test program [1]. Our main focus is on engineering novel features for statistical tests and demonstrating the

importance of feature engineering for effectively capturing test escapes by statistical tests. Specifically, we develop features based on how a chip's measurements deviate from the means of a set of normal chips and how a chip's measurements deviate from the spatial patterns on a wafer. We then transform the features to a canonical space in which the separation between normal chips and test escapes of the projected data is maximized. The multivariate statistical approach based on these features incorporates both the inter-test-item correlations and the spatial correlations, and applying statistical tests based on the transformed features achieves significant runtime reduction based on standard classification algorithms. The proposed flow can be easily extended to include more sets of features and applied to a wide range of products.

To screen potential test escapes, one technique proposed by the Automotive Electronics Council is the part average testing (PAT) [4]. For some suggested electrical tests, PAT compares the measurement of a query chip with the mean of a set of normal chips and discard the query chip if it is more than 6-σ away from the mean. To address PAT's limitation of evaluating individual test item only and ignoring the multivariate relation among test items, several other studies proposed multivariate screening approaches that incorporate the inter-test-item correlations.

O'Neill [5] applied outlier analysis with principal component analysis (PCA) on sets of correlated test items. Sumikawa *et al.* [2] extended O'Neill's work with sophisticated model and test selection schemes and developed a preemptive and a reactive approach, depending on whether known field returns were given. Butler *et al.* [6] successfully demonstrated burn-in minimization by a collection of

multivariate analysis. Chen *et al.* [3] showed various data mining techniques on final test data to predict system level test (SLT) failures.

In addition to inter-test-item correlations, there exist spatial correlations among dies on the same wafer. Stine *et al.* [7] modeled and decomposed spatial variations into four components: wafer-level variation, die-level variation, wafer-die interactions, and residuals. In capturing the spatial patterns with only a small amount of samples, Li *et al.* [8] proposed a virtual probe (VP) technique and Kupp *et al.* [9] proposed an estimation with a Gaussian process model. Nahar *et al.* [10] and Riordan *et al.* [11] used the spatial correlation of neighboring dies for defect prediction. Sumikawa *et al.* [12] identified abnormal wafers based on the spatial patterns of tests.

Taking into account the multiple correlations in test data, we also investigate a data transformation technique based on multivariate analysis of variance (MANOVA). MANOVA has been used in various fields to analyze the difference in the means of features between populations of samples [13, 14]. Based on MANOVA, a canonical analysis can be applied on the samples to form a set of canonical variables which are linear combinations of the original test measurements. The linear combinations are chosen such that the first canonical variable achieves the maximum separation between populations, the second canonical variable achieves the maximum separation between populations subject to it being orthogonal to the first canonical variable, and so on. Essentially MANOVA shows if there is a significant difference in the means between populations, and the canonical analysis could identify combinations of the variables to maximize the separation between populations. In this chapter we utilize the canonical analysis

to project the features into a canonical space for further statistical tests, which can more easily screen out test escapes with standard classification methods.

The rest of the chapter is organized as the following: Section 2.2 illustrates how we develop features that represent different characteristics of a chip. Section 2.3 introduces the canonical analysis to further transform the features. The application of the proposed statistical tests is described in Section 2.4, and Section 2.5 shows experimental results on production test data. Section **??** concludes the chapter.

2.2 Feature Development

In our research, we use the *residual vector* of each chip as the base of input features for statistical analysis. In general, each chip with N test measurements can be characterized by an $N \times 1$ residual vector $\boldsymbol{r}$:

$$\boldsymbol{r} = \boldsymbol{x_m} - \boldsymbol{x_e} \tag{2.1}$$

where $\boldsymbol{x_m}$ is an $N \times 1$ vector of the measured values and $\boldsymbol{x_e}$ is an $N \times 1$ vector of the estimated values.

The residual vector represents how the measurement values of a chip deviate from its estimated values. There are two aspects we can engineer such feature. Choosing different estimated values to calculate the residual vector may reveal different test escapes, and transforming the residual vector to another space may enhance the performance of the classifier.

We first explore the choices of estimated values for calculating the residual

vector. The transformation of test data to another space will be discussed in Section 2.3.

2.2.1 Measurement Mean

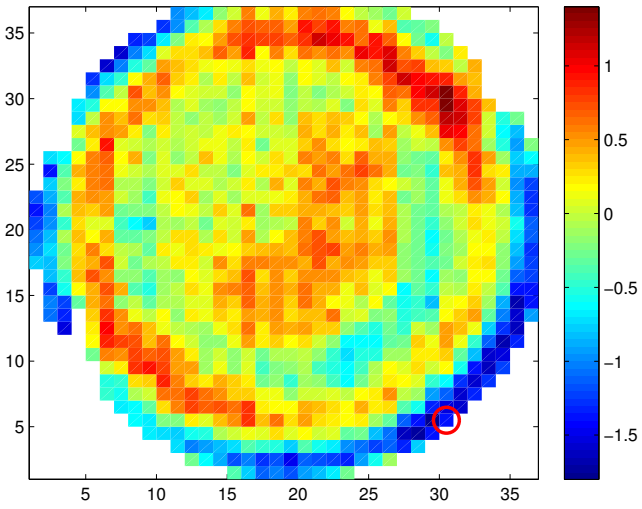
It has been shown that some test escapes differ from normal chips in their test measurement values relative to the measurement means [3]. Fig. 2.1 shows two wafer maps whose measurement values are standardized to the z-score by

$$z = \frac{x - \mu}{\sigma} \tag{2.2}$$

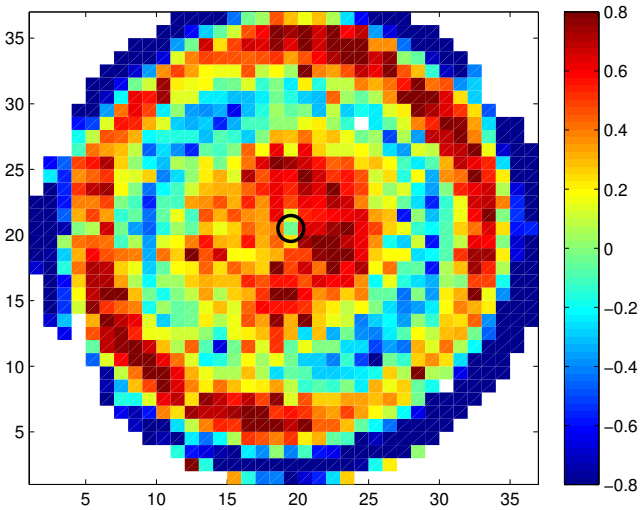
where x is the measurement value, and μ and σ are the mean and standard deviation of the measurements for chips on the same wafer, respectively. In Fig. 2.1a the circled chip is a test escape with a relatively abnormal feature value based on the difference between its measurement value and the measurement mean of the entire wafer. Note that Fig. 2.1a shows only one test item and thus only one out of N features of the chips. There are other chips in the same dark blue region, which are as outlying as the circled chip is for this test item, but in the multivariate analysis they will not necessarily be classified as escapes.

2.2.2 Spatial Pattern

Spatial patterns have been observed on wafers in many test items [7, 15, 16, 17, 12]. Fig. 2.1b shows a test escape which would be considered abnormal based on spatial pattern analysis. A wafer's spatial patterns for some test items are the results of systematic variations which exist even without any additional man-



(a) A test escape that is abnormal with respect to measurement mean



(b) A test escape that is abnormal with respect to spatial pattern

Figure 2.1: Examples of test escapes which are considered abnormal with respect to different estimated values.

ufacturing imperfections. Taking into account a test item's spatial pattern and using the unique predicted value at each die location (derived from the test item's learned spatial pattern) as the estimated value, we effectively eliminate the systematic variations and incorporate only the effects of other variations (including random variations) in the residual vector. In other words, with this revised residual vector which takes into account test items' spatial patterns, we more accurately capture the noises in the wafer map images. We employ *bilateral filtering*, a well-developed filtering technique in image processing, to denoise a wafer map and retrieve a systematic spatial pattern of a test item.

Bilateral filtering [18] is a non-linear filtering technique which extends the concept of Gaussian filtering to weight coefficients based on both relative spatial distance and pixel intensity difference. Pixels that are spatially close but have significant difference in intensity will have smaller weights, while pixels that are a bit farther apart but very similar in intensity will have larger weights. Therefore, the sharp edges of an image can be preserved and the noises are more likely to be filtered. There are two kernels for evaluating the weights of the neighboring pixels. The *domain kernel* K_d evaluates the weights based on the spatial distance of pixels. The *range kernel* K_r evaluates the weights based on the pixel intensity difference.

Given the original image I, pixel coordinates x, and the filter window Ω, the filtered image I_f is defined as

$$I_f(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) K_r(\|I(x_i) - I(x)\|) K_d(\|x_i - x\|) \tag{2.3}$$

where

$$W_p = \sum_{x_i \in \Omega} K_r(\|I(x_i) - I(x)\|) K_d(\|x_i - x\|) \tag{2.4}$$

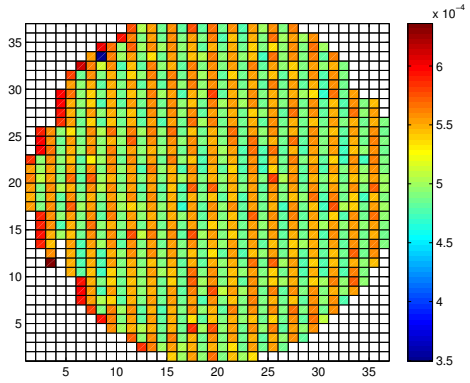
For a $P \times Q$ wafer map with values ranging from v_{min} to v_{max}, we choose a Gaussian function with $\sigma = min(P, Q)/16$ for K_d, a Gaussian function with $\sigma = (v_{max} - v_{min})/10$ for K_r, and the whole wafer map as the filter window Ω.

Fig. 2.2 shows the result of applying bilateral filter to one test item on a wafer map. Fig. 2.2a shows the original measurement of the test item. The residual of the measurement with respect to the wafer mean is shown in Fig. 2.2b, in which the spatial pattern is preserved. The residual of the measurement with respect to the bilateral filtered wafer map is shown in Fig. 2.2c, in which the spatial pattern in the original measurement is eliminated and the residuals better represent abnormality with respect to the spatial pattern.

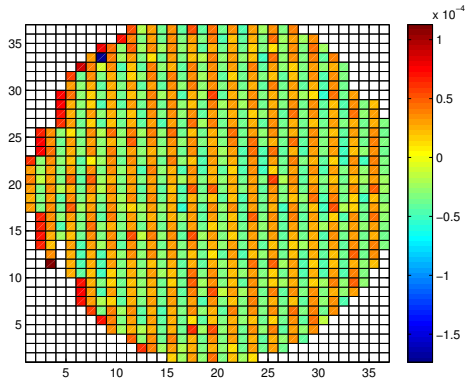
Note that the circled test escape in Fig. 2.1a is abnormal considering its relatively large measurement value, but it is perfectly normal if we take into account the overall spatial pattern. The circled test escape in Fig. 2.1b is abnormal in the spatial pattern, but its measurement value is actually very close to the mean of the wafer. Therefore, each of the two test escapes shown in Fig. 2.1 can only be uniquely identified as abnormal, or potential test escape, by one of the two choices in selecting the estimated values for calculating residual vectors.

2.3 Feature Transformation

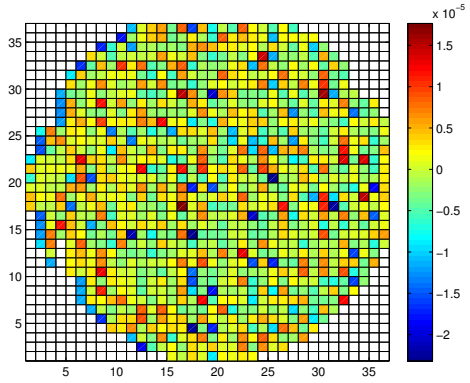
Besides exploring two different choices of the estimated values for calculating the residual vectors to enrich the input features, projecting these features into



(a) Original measurement



(b) Residual with respect to the mean of the measurement



(c) Residual with respect to the bilateral filtered measurement

Figure 2.2: An example of residuals with respect to different estimated values of one test item.

different spaces before applying statistical tests may help improve the performance of classifiers. We introduce canonical analysis based on multivariate analysis of variance (MANOVA) to transform data into a canonical space in which the data of test escapes and normal chips are maximally separated in the first few dimensions.

Multivariate analysis of variance (MANOVA) [13] is a technique that, given g populations of samples in an N-dimensional space, compares the mean vectors of the populations and investigates which mean components differ significantly. Given the populations of samples:

$$\begin{aligned} &Population\ 1 : \boldsymbol{x}_{11}, \boldsymbol{x}_{12}, ...\boldsymbol{x}_{1n_1} \\ &Population\ 2 : \boldsymbol{x}_{21}, \boldsymbol{x}_{22}, ...\boldsymbol{x}_{2n_2} \\ &\qquad\vdots \\ &Population\ g : \boldsymbol{x}_{g1}, \boldsymbol{x}_{g2}, ...\boldsymbol{x}_{gn_g} \end{aligned} \tag{2.5}$$

where $\boldsymbol{x}_{ij}$ is an $N \times 1$ vector of the jth sample in the ith population, and n_i is the number of samples in the ith population.

Each observation $\boldsymbol{x}_{ij}$ can be decomposed into three components: overall sample mean $\bar{\boldsymbol{x}}$, the population effect $(\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})$, and the residual $(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i)$, and

$$\boldsymbol{x}_{ij} = \bar{\boldsymbol{x}} + (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}) + (\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}}_i) \tag{2.6}$$

Subtracting $\bar{\boldsymbol{x}}$ from both sides of (2.6) and summing the cross products over i and

j yields

$$\begin{aligned} &\sum_{i=1}^{g}\sum_{j=1}^{n_i}(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}})(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}})' \\ =&\sum_{i=1}^{g} n_i(\bar{\boldsymbol{x}}_i-\bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i-\bar{\boldsymbol{x}})' + \sum_{i=1}^{g}\sum_{j=1}^{n_i}(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}}_i)' \end{aligned} \tag{2.7}$$

or expressing it as:

$$\boldsymbol{S} = \boldsymbol{B} + \boldsymbol{W} \tag{2.8}$$

where

$$\begin{aligned} \boldsymbol{S} &= \sum_{i=1}^{g}\sum_{j=1}^{n_i}(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}})(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}})' \\ \boldsymbol{B} &= \sum_{i=1}^{g} n_i(\bar{\boldsymbol{x}}_i-\bar{\boldsymbol{x}})(\bar{\boldsymbol{x}}_i-\bar{\boldsymbol{x}})' \\ \boldsymbol{W} &= \sum_{i=1}^{g}\sum_{j=1}^{n_i}(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}}_i)(\boldsymbol{x}_{ij}-\bar{\boldsymbol{x}}_i)' \end{aligned} \tag{2.9}$$

Eq. (2.8) shows that the total variance $\boldsymbol{S}$ is the sum of the between-population variance $\boldsymbol{B}$ and the within-population variance $\boldsymbol{W}$. After such decomposition, MANOVA investigates if there exists significant difference between the population mean vectors using metrics based on $\boldsymbol{B}$ and $\boldsymbol{W}$. For example, if the Wilks' lambda

$$\Lambda = \frac{|\boldsymbol{W}|}{|\boldsymbol{B}+\boldsymbol{W}|} \tag{2.10}$$

is too small, we can conclude that there exists significant difference between the populations.

After MANOVA, the canonical analysis suggested in [14] could be used to create a set of canonical variables which are the linear combinations of the original variables. The criteria for choosing the linear combinations are that the first canonical variable should exhibit the maximum separation between the populations, the second canonical variable should be orthogonal to the first canonical variable while also exhibit maximum separation between the populations, and so on.

The process of generating the canonical variables in canonical analysis is similar to generating the principal components (PCs) in principal component analysis (PCA) [13]. PCA is a feature reduction technique that generates a set of new features, named principal components, which are mutually orthogonal and are ordered by the amount of variability in the data each PC explains. Given a set of data in matrix $\boldsymbol{X}$, where rows of $\boldsymbol{X}$ represent observations, and columns of $\boldsymbol{X}$ represent variables, PCA creates the first PC, the linear combination of variables that can maximally explain the multivariate variability in $\boldsymbol{X}$, using the eigenvector of the covariance matrix $\boldsymbol{X}'\boldsymbol{X}$ with the largest eigenvalue. The second PC is the eigenvector of $\boldsymbol{X}'\boldsymbol{X}$ with the second largest eigenvalue, which explains the maximum variability of $\boldsymbol{X}$ subject to it being orthogonal to the first PC. In canonical analysis, canonical variables are chosen based on the ability of explaining the ratio of the between-population variance $\boldsymbol{B}$ over the within-population variance $\boldsymbol{W}$, so that in the first canonical variable the populations are maximally separated. Therefore, the first canonical variable is derived as the eigenvector of $\boldsymbol{W}^{-1}\boldsymbol{B}$ with the largest eigenvalue, the second canonical variable is chosen as the eigenvector of $\boldsymbol{W}^{-1}\boldsymbol{B}$ with the second largest eigenvalue, and so on.

Let $\boldsymbol{E}$ be the matrix whose first column is the first eigenvector of $\boldsymbol{W}^{-1}\boldsymbol{B}$, the second column is the second eigenvector, and so on, and each eigenvector is scaled such that the within-population variance of the canonical variable is 1. A new data set $\boldsymbol{Y}$ can be projected to the canonical space by

$$\boldsymbol{Y_p} = \boldsymbol{Y_c}\boldsymbol{E} \tag{2.11}$$

where $\boldsymbol{Y_c}$ is $\boldsymbol{Y}$ with columns centered by subtracting their means and $\boldsymbol{Y_p}$ is the projected data set.

Both PCA and canonical analysis project data to another space by linear transformation, but the objectives of the transformations are quite different. PCA orders the created variables according to the amount of the variability in the data the variables can explain, while canonical analysis orders the created variables according to the amount of the between-population variance over the within-population variance the variables can explain. Both PCA and canonical analysis can be used for feature reduction. With a limited number of created variables which is smaller than the dimension of the original data, PCA preserves the variability of the data and canonical analysis preserves the separation between the populations of the data. Because the separation between populations is compacted into a small number of variables, the populations of data will be maximally separated in the space formed by the first few canonical variables, and a standard classification algorithm can much more easily classify the samples.

In multivariate statistical analysis, the canonical analysis can be regarded as canonical correlation analysis [14, 19, 20] between the dependent variables and

some dummy variables. The description above for the process of deriving the canonical variables is similar to Fisher's linear discriminant analysis (LDA) [21], in which the canonical variables are known as discriminants. Applying the canonical analysis to our application, we categorize the normal chips as one population and the test escapes as the second population, and create the canonical variables by linear combinations of the test items.

2.4 Test Methodology

Based on the feature engineering scheme discussed in Sections 2.2 and 2.3, we propose to use two distinct sets of features. The first set of features are the residual vectors with measurement means as the estimated values, followed by the transformation to the canonical space. The second set of features are the residual vectors which use predicted values from the learned spatial patterns as the estimated values, followed by the transformation to the canonical space.

The two sets of features can be utilized in two ways. First, each set of features is used as the input features for one classifier - determining if the chip under test belongs to the normal population or test escape population, and the classifiers together form a series of statistical tests. A second possible way of utilizing the features is to include all sets of features to form a single comprehensive statistical test. The exemplar test flows are demonstrated in this section.

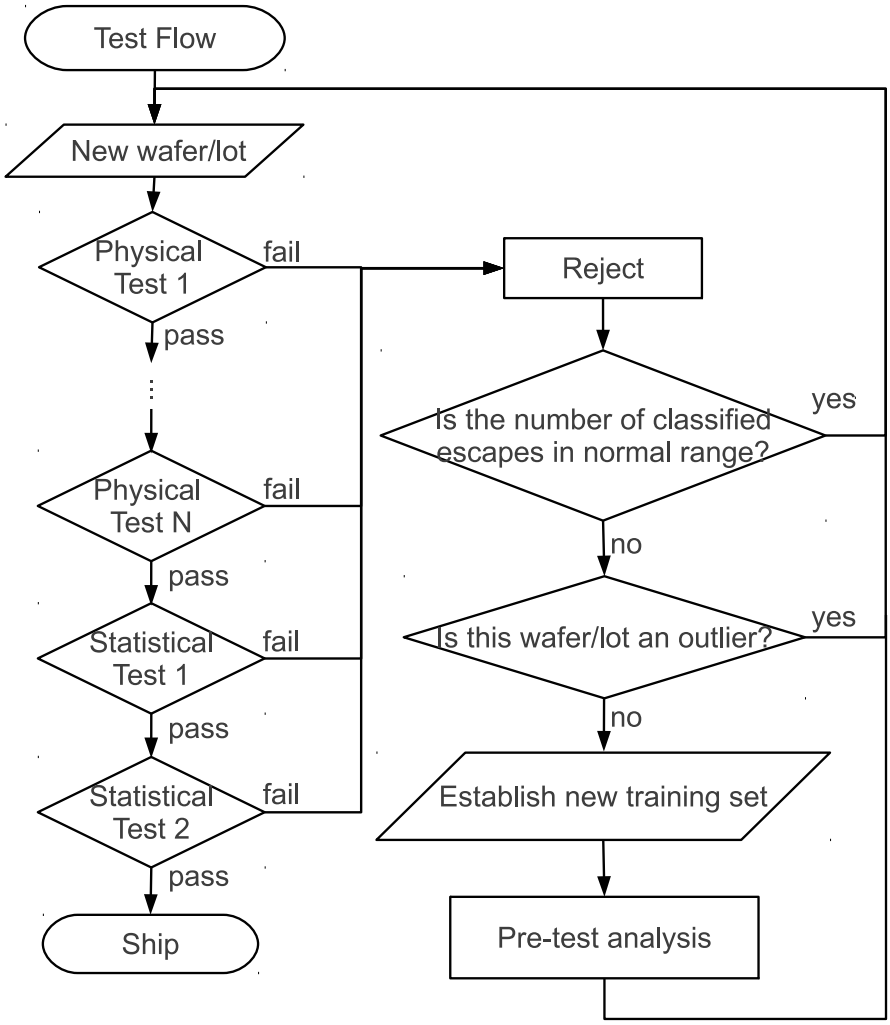


Figure 2.3: Test flow with sequential statistical tests

2.4.1 Classifier

We use the *C-support vector classification (C-SVC)* algorithm provided by the SVM library LIBSVM [22] as our classifier for separating test escapes and good chips. Given the fact that the two classes of samples are very imbalanced (i.e. the number of good chips is much greater than the number of test escapes) in a practical training set, one can set a much higher weight for the class of test escapes to force the algorithm to always find a model that correctly identifies escapes [23]. The guideline for our classification, however, is to screen out as many escapes as possible subject to the constraint of limiting the yield loss to a very small number (say, less than 0.001%). Based on this guideline, each class in the C-SVC is given the same weight in our experiment to allow a thorough search for a model with the maximum number of correctly identified escapes while minimizing the yield loss to a level very close to 0.

2.4.2 Pre-test Analysis

To start the proposed statistical tests, the canonical variables and the C-SVC models based on the search for the optimal combination of parameters [22] need to be generated based on a set of training chips, and the optimal C-SVC model needs to be selected based on a set of validation chips. The training/validation set of good chips should be sampled across several lots and wafers, and the distribution of measurements in each lot should be checked for uniformity to validate that the training/validation set indeed properly represents a good-chip population. Some field returns (or known test escapes that pass the test program) are also required for finding the canonical variables and the classification models. Section 2.5 will

show that only a very small ratio of returns/known test escapes is required in order to find a canonical space for achieving significant runtime reduction while preserving the discriminating power between the test escapes and the normal population.

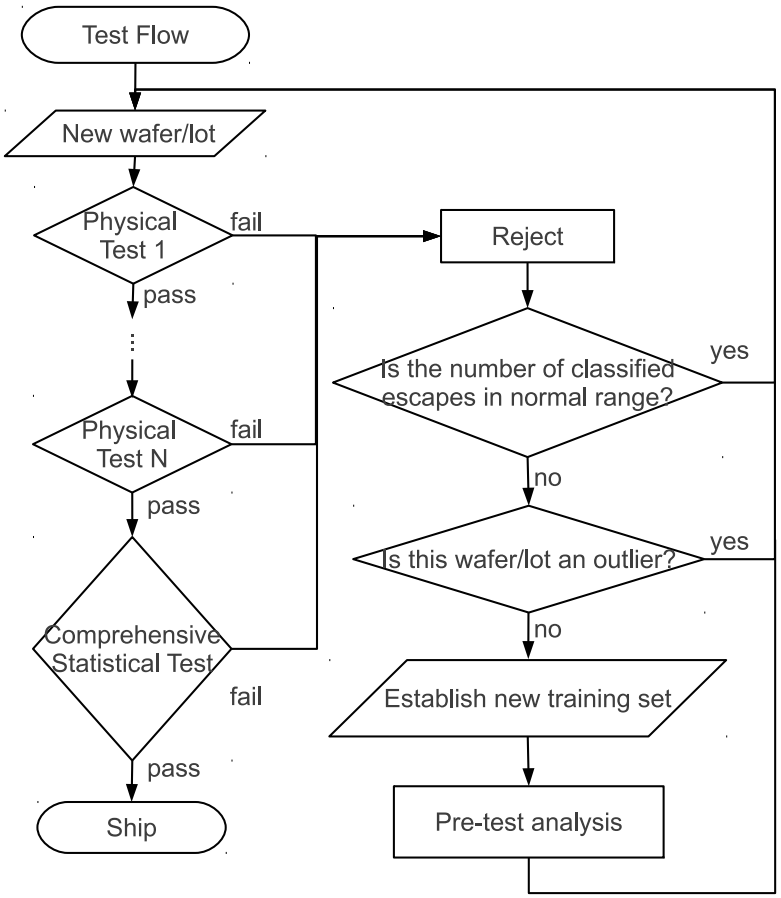


Figure 2.4: Test flow with a comprehensive statistical test

2.4.3 Test Application

An exemplar test flow of the proposed statistical tests, applied to each wafer/lot, is shown in Fig. 2.3. These statistical tests are performed after all physical tests

are executed, and serve as additional *rejectors* which reject some of the bad chips that escape all physical tests. While in this chapter we suggest two specific statistical tests only, additional statistical tests can be developed and applied based on more new features.

A modification from Fig. 2.3 is shown in Fig. 2.4, in which the sequential statistical tests are replaced with one single comprehensive statistical test. With the application of canonical transform, the useful information in separating the test escapes and the normal population in all generated features is incorporated into the comprehensive test. An experimental comparison on the two exemplar flows will be made in Section 2.5.

Given any training set, it is possible that the manufacturing process drifts over time such that the training set is no longer sufficiently representative for the later data. Therefore, if the proposed statistical tests report an abnormally large number of test escapes for a wafer/lot under test, it could be due to such temporal process variation and thus the wafer/lot should be analyzed for the actual cause. If the wafer/lot is diagnosed as an outlier wafer/lot, we can conclude that the training set still effectively represents the population of good chips, and the test flow can continue to the next wafer/lot without a new training set. Otherwise, a new training set should be established and the corresponding canonical variables and C-SVC models should in turn be derived.

2.5 Experimental Result

To validate the proposed methods, the continue-on-fail production test data of a high volume commercial product was first preprocessed to remove confidential information while accurately preserving the information critical to the evaluation. The data set contains more than 1200 wafers with more than 200 parametric test measurements captured by the production test program for each die, and have more than 700 dies per wafer. In the following discussion we use N to denote the number of parametric test measurements.

Since there is no actual test escape information in this data set, we first introduce how to *emulate* test escapes for our evaluation, and then validate our proposed methodology based on the data set.

2.5.1 Data Setup

In a test program, chips with measurements beyond the test limits are rejected as faulty chips, and those pass the test program but fail later at the system level or in the field (field returns) are test escapes. Without actual test escape information, we identified a set of bad chips as *emulated* test escapes which meet the following criterion: among the over 200 measurements, only one measurement did not fall within its test limits and its violation to the spec was marginal. We therefore *hid* this failing measurement which rejected these faulty chips - pretending that each of these bad chips still passed all physical tests in the test program and is treated as a test escape. To hide this measurement which failed a chip, we replaced its value by a normal value well within the test limits such that the resulting

feature value (i.e. an element of the chip's residual vector) is equal to the median of all good chips. After such manipulation, these faulty chips now have similar characteristics as test escapes: passing the complete test program, but with some intrinsic defects. We can then evaluate if the measurements of all-but-one test items which did not violate the test limits can expose those *emulated* test escapes in the proposed statistical tests.

The goal of our analysis is to screen test escapes based on the subtle differences in the non-failing measurements, so we only considered those faulty chips with a very small number of failing measurements and insignificant violations to the test specs as emulated test escapes. Faulty chips which fail many test items and/or have significant violations to the test specs are more likely to be catastrophic failures. Such catastrophic failures should have very revealing and differentiable features derived from the non-failing measurements, and including those catastrophic failures could result in an overly optimistic conclusion of the experiment. Therefore, we excluded faulty chips with more than one failing measurements or with only one failing measurement but its violation to the spec is significant from our analysis.

2.5.2 Sequential Rejectors

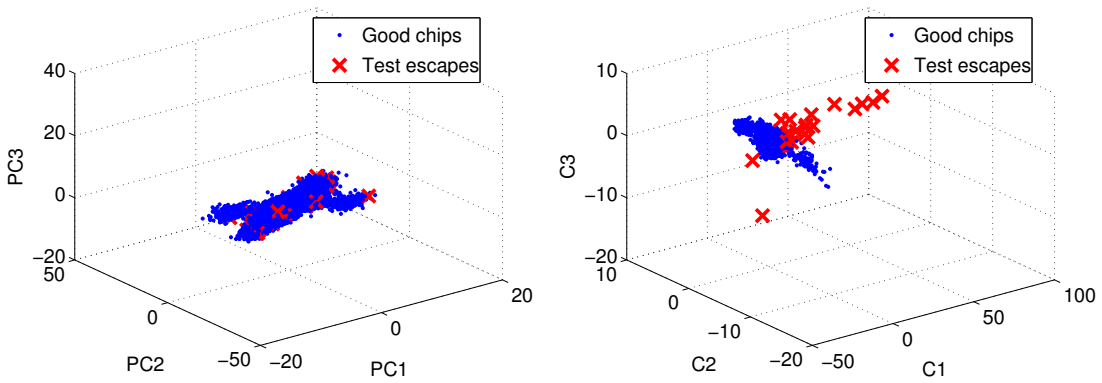
In pre-test analysis, test data from 200 wafers were used as the training set for finding the canonical variables and generating C-SVC models based on the search for the best combination parameters. Test data from another 300 wafers were used as the validation set for selecting the best C-SVC model with an acceptable level of yield loss.

For simplicity and clarity, we use the following notations for the different residual vectors:

- Feature Set F_m: Residual vectors derived using measurement means as the estimated values
- Feature Set F_s: Residual vectors derived using predicted values based on the learned spatial patterns as the estimated values

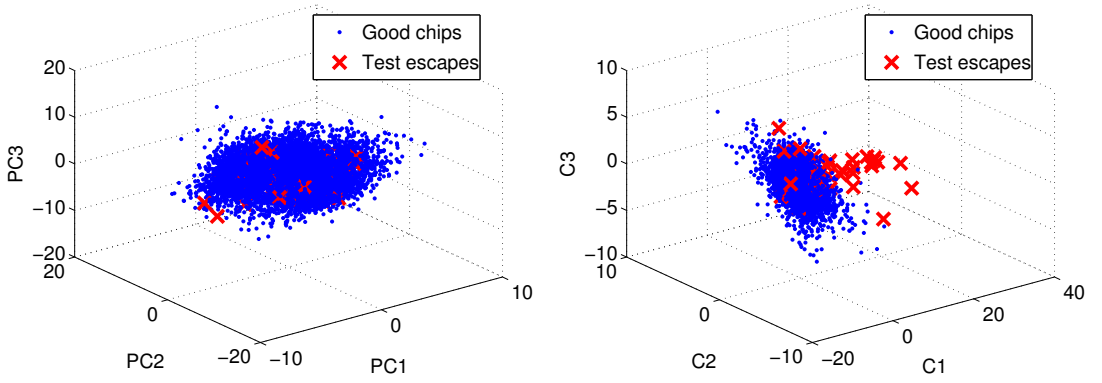
After the canonical space is found based on the training set, the residual vectors of the validation set are transformed into the canonical space by Eq. (2.11). For comparison, we also transform these residual vectors into the PC space by PCA. Fig. 2.5 illustrates the feature set F_m transformed into the PC space and the canonical space respectively (with respect to the first three variables created in each of these two analyses). Fig. 2.6 shows feature set F_s transformed into the PC space and the canonical space respectively. For better visualization, Fig. 2.5 and Fig. 2.6 show only a subset of good chips and emulated test escapes (they were randomly sampled from the validation set while maintaining the original ratio of good chips vs. test escapes). It is very clear that the test escapes are much more separable from the normal population in the canonical space than in the PC space for both F_m and F_s.

Fig. 2.7 shows the relative operating characteristics (ROC) diagram of C-SVC's performance based on F_m and F_s in the original space, the PC space, and the canonical space. The horizontal axis shows the yield loss rate and the vertical axis shows the test escape identification rate, which are the *false positive rate* and the *true positive rate* respectively in our classification problem. Using all



(a) PC space (b) Canonical space

Figure 2.5: The distributions of good chips/test escapes in different feature spaces of F_m.



(a) PC space (b) Canonical space

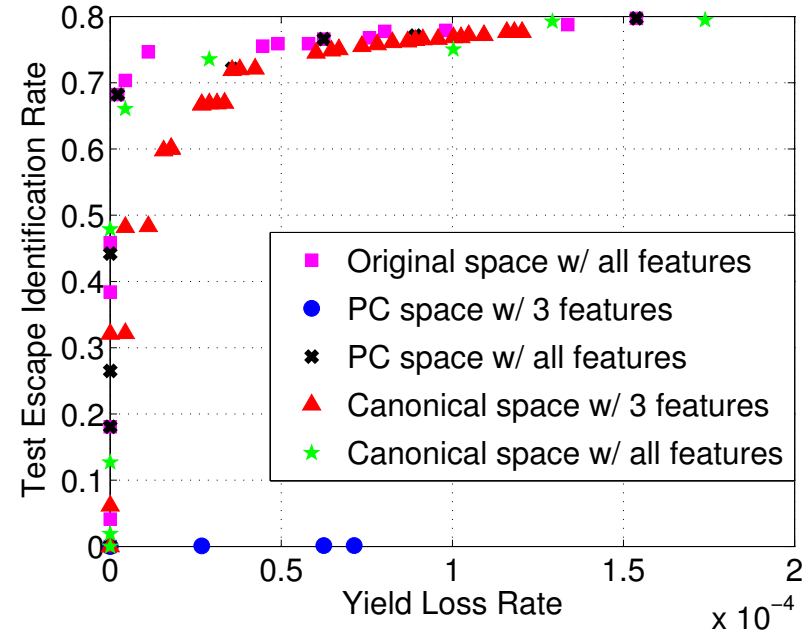
Figure 2.6: The distributions of good chips/test escapes in different feature spaces of F_s.

features in the three spaces results in similar classification performances because the discriminating information is preserved in the linear transformation to a new space. When only three features are used for the purpose of feature reduction, using the first three canonical variables as input features results in significantly greater classification accuracy than using the first three PCs, which explain only 25% and 21% of the variability in F_m and F_s respectively.

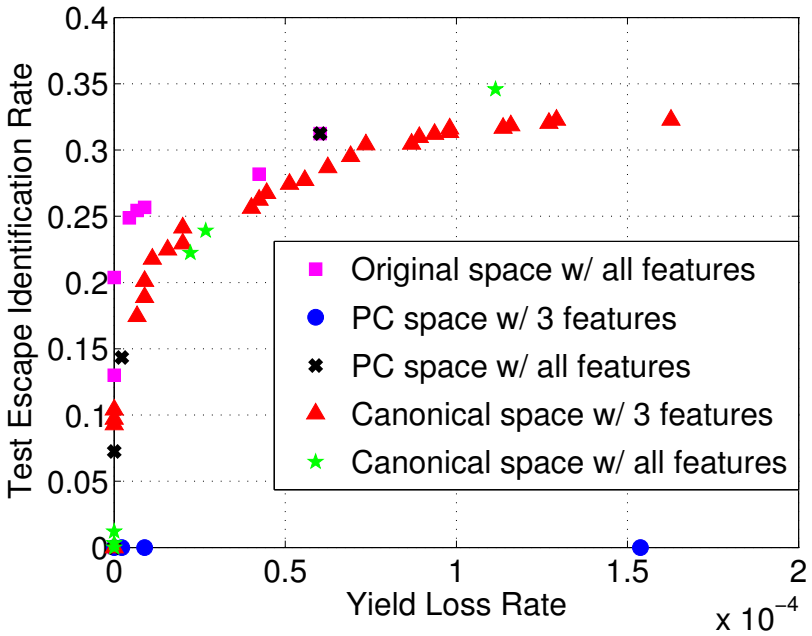
Table 2.1 shows the ratio of test escapes identified by C-SVC in the N-dimensional original feature space, 3-dimensional PC space, and 3-dimensional canonical space, with a limit of 0.001% yield loss for all three cases. The first column shows the ratio of test escapes identified only by F_m and not detectable by F_s. The second column shows the ratio of test escapes identified only by F_s and not detectable by F_m, and the third column shows the ratio of the union of the test escapes identified by F_m and F_s. C-SVC in the 3-dimensional PC space cannot identify any of the test escapes given the very low yield loss rate limit, while C-SVC in the 3-dimensional canonical space achieves a lower but still significant ratio of identified test escapes than C-SVC with all the features in the original space.

For a fixed yield loss budget (0.001% in this experiment), the learned C-SVC models based on different sets of features can identify unique sets of escapes. In the canonical space, there are 11.0% of the escapes that can be identified based on both F_m and F_s, while the classifications based on F_m and F_s identify unique sets of 37.2% and 9.1% of the escapes, respectively.

To better visualize a feature set's ability of uniquely identifying test escapes, Fig. 2.8 shows the distribution of identified test escapes in the canonical space of



(a) ROC of C-SVC based on F_m



(b) ROC of C-SVC based on F_s

Figure 2.7: The ROC diagram for C-SVC based on F_m and F_s in three different spaces.

Table 2.1: Percentage of Test Escapes Detected by C-SVC in Three Different Spaces Based on the Validation Set

	Detected by Feature Set		
	F_m only	F_s only	F_m or F_s
C-SVC in N-D original space[a]	45.7%	3.2%	71.4%
C-SVC in 3-D PC space	0%	0%	0%
C-SVC in 3-D canonical space	37.2%	9.1%	57.3%

[a] N: Number of Test Measurements (> 200)

F_m. It is clear that F_s reveals some test escapes that are close to the normal population in the canonical space derived from F_m, which C-SVC in this space cannot correctly classify without incurring additional yield loss. Therefore, it is important to incorporate both sets of features to screen test escapes more effectively.

2.5.3 Comprehensive Test

In addition to applying canonical transform to F_m and F_s separately, we can apply canonical transform to F_m and F_s together, considering them as a single feature set for the classification problem. In this case, the proposed framework becomes even more extensible as one can generate many possible features and input them all into the canonical transform without knowing which sets of features are more suitable for which product. A general feature set can be developed and applied to different products efficiently since canonical transform automatically generates the most discriminating features out of all possible features for each product. The test flow is shown in Fig. 2.4 in Section 2.4.

Fig. 2.9 shows the distribution of good chips and test escapes in the canonical space derived from $F_m \cup F_s$, i.e. a canonical variable is a linear combination of all

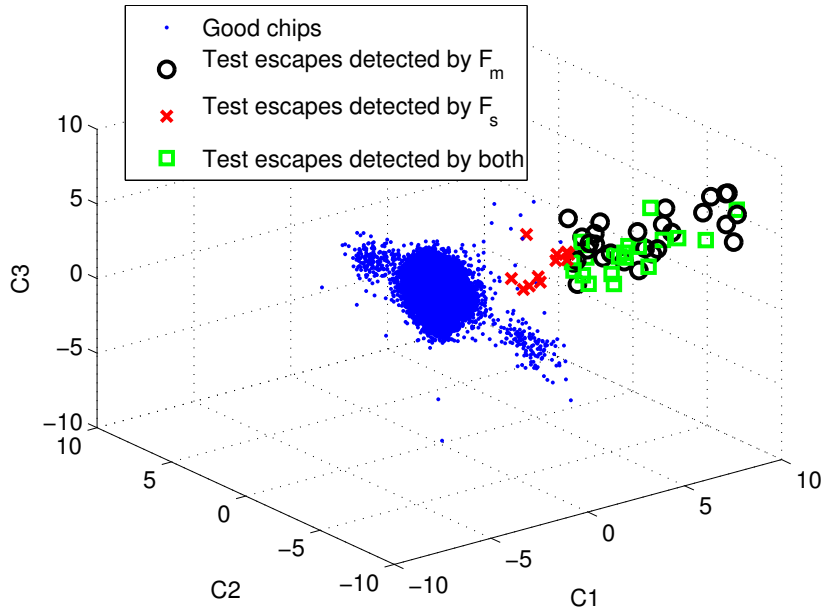


Figure 2.8: The distributions of good chips and test escapes detected by the two sets of features in the canonical space of F_m.

features in F_m and all features in F_s. There is a clear separation between the good chips and the test escapes. Fig. 2.10 shows the ROC curves of using 3 canonical variables derived from F_m, F_s, and $F_m \cup F_s$, as the input features for C-SVC. When the yield loss rate is very low ($< 0.001\%$), using the canonical variables derived from $F_m \cup F_s$ identifies 64.6% of the test escapes, which is higher than using the canonical variables derived from F_m or F_s alone. Compared with the results in Table 2.1, using canonical variables derived from $F_m \cup F_s$ identifies more test escapes than the union of the test escapes identified using the canonical variables derived from F_m and the canonical variables derived from F_s. In other words, using a single comprehensive test derived from all generated features achieves greater accuracy than using a sequence of rejectors, each of which is derived from

a unique set of features.

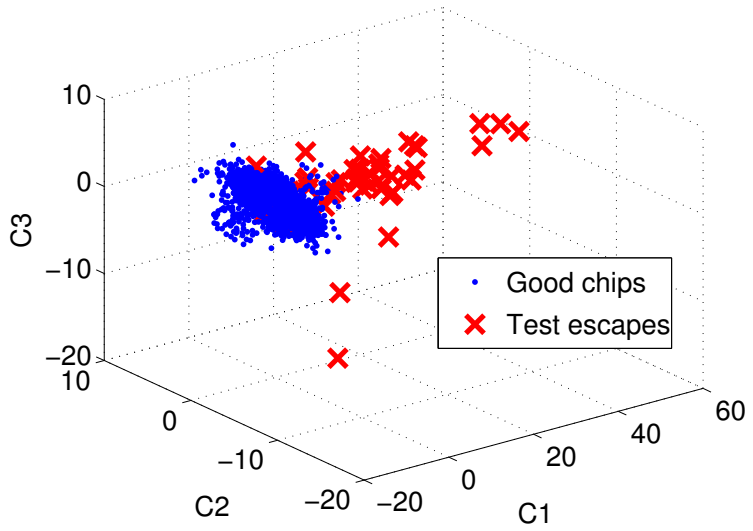


Figure 2.9: The distributions of good chips/test escapes in the canonical space derived from $F_m \cup F_s$.

2.5.4 Test Application

The testing set of our data includes more than 700 wafers containing more than 500K chips. The percentage of detected test escapes based on the testing set, given a yield loss budget of 0.001%, is shown in Table 2.2. The classification performance is greatly enhanced in the canonical space than that in the PC space. Using the canonical variables derived from $F_m \cup F_s$ also results in much better accuracy than using the canonical variables derived from F_m or F_s alone. Note that in the original space there are N dimensions in F_m and F_s, $2N$ dimensions in $F_m \cup F_s$, where N is the number of test measurements.

The following two tables show the runtimes executed on an Intel Xeon Quad-

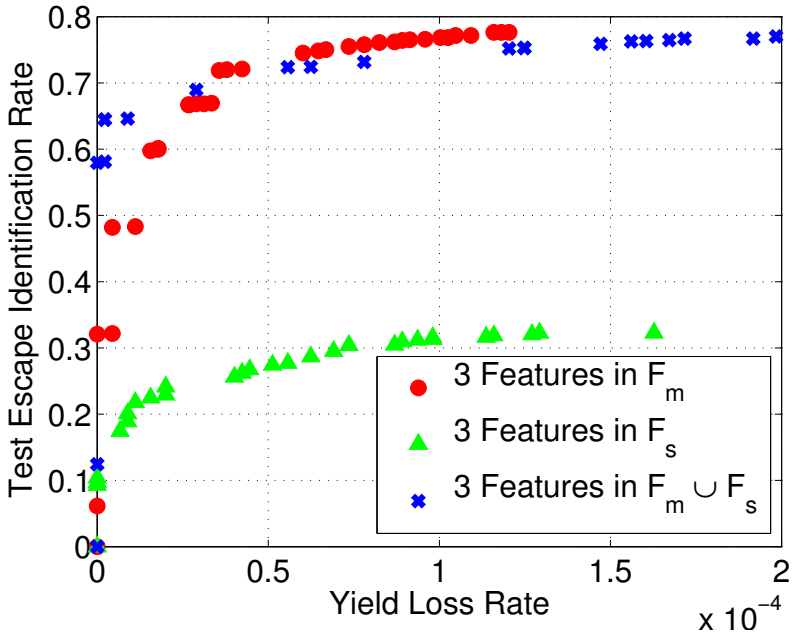


Figure 2.10: The ROC diagram for C-SVC in canonical space with 3 features derived from different feature sets.

Table 2.2: Percentage of Test Escapes Detected by C-SVC in Three Different Spaces Based on the Testing Set

	Feature Set		
	F_m	F_s	$F_m \cup F_s$
C-SVC in original space	64.4%	23.6%	67.1%
C-SVC in 3-D PC space	0%	0%	0.07%
C-SVC in 3-D canonical space	43.4%	17.1%	61.9%

core 3.6GHz system. The data is based on the comprehensive test, in which the canonical variables are derived from $F_m \cup F_s$.

The runtime for canonical transform and PCA to derive the transform matrix based on the training set and to apply the transform on the testing set is shown in Table 2.3. While it takes slightly longer to derive the transform matrix for canonical transform during the training process, the time for applying the transform is very close for both transforms.

Table 2.3: Runtime of Deriving/Applying Transform Matrix for Canonical Transform and PCA Based on Training/Testing Set

	Transform	
	Canonical Transform	PCA
Derivation	5.55 s	3.56 s
Application	0.75 s	0.78 s

Table 2.4 shows the runtime of training the C-SVC model based on the training set and applying the C-SVC model to the testing set. Training in the canonical space is much easier because most of the separating power in the data is compacted into the 3 input features for C-SVC. A runtime reduction of 63X and 29X is achieved for training and applying the model in the canonical space. In application of the proposed statistical test to the test flow, it takes 0.75s + 8.41s to transform the features and apply the model to more than 700 wafers, resulting in less than 0.013s additional test time per wafer.

Table 2.4: Runtime of Training/Applying C-SVC Model in Three Different Spaces Based on Training/Testing Set

	Feature Space		
	$2N$-D Original	3-D Canonical	3-D PC
Training	31.27 s	0.50 s	3.42 s
Application	242.16 s	8.41 s	74.87 s

2.5.5 Another Experimental Scenario

The emulated data set based on the previous description contains 3500PPM of test escapes, a good fraction of which are detected by a very small number of test items. In order to capture a more realistic scenario for which each test item detects only a small number of emulated test escapes, we removed some escapes from the original emulated test escape population (of 3500PPM) so that each test item only detects a limited number of escapes in the resulting test escape population. Specifically, we identified those test items that hiding each of them would result in greater than 50PPM in the original test escape population. We then removed those emulated escapes detected by these test items, resulting in a reduced emulated test escape population of 600PPM.

Based on the data set with a reduced number of test escapes, the ROC curves for C-SVC with $F_m \cup F_s$ as the input features are shown in Fig. 2.11. Note that in this case, C-SVC in the canonical space, even with only the first 3 dimensions, achieves greater classification accuracy than C-SVC in the original space with all $2N$ features. For a yield loss rate limited to 0.001%, Table 2.5 shows the test escape identification rate and the corresponding runtime for applying the model. In this data set, effectively compacting the separation between classes into the

very few dimensions allows more effective and efficient classification for C-SVC than that in the original space with much more dimensions.

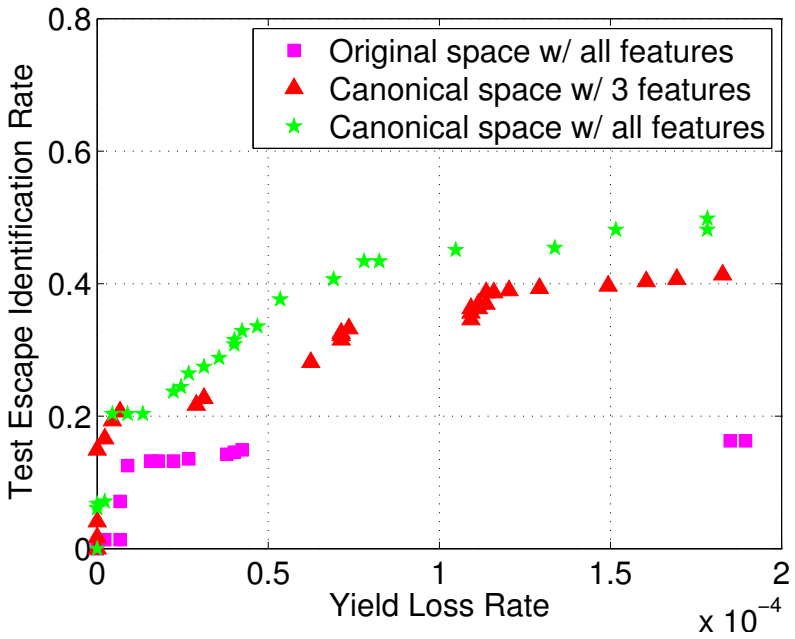


Figure 2.11: The ROC diagram for C-SVC based on $F_m \cup F_s$ in the original space and in the canonical space.

Fig. 2.12 shows the ROC diagram of classification based on 3 canonical variables derived from F_m, F_s, and $F_m \cup F_s$. Given the yield loss rate limited to 0.001%, the ratio of identified test escapes are 14.6%, 0.7%, and 16.3%, respectively. While the identification rate drops for the three cases compared with that in the original data set, classification based on the canonical variables derived from $F_m \cup F_s$ still achieves better accuracy than using the canonical variables derived from F_m or F_s alone, and the identification rate of test escapes is still significant under the constraint of a close-to-zero limit on yield loss rate.

Table 2.5: Test Escape Identification Rate and Runtime for Applying the Models with the Yield Loss Rate Limited to 0.001%, based on $F_m \cup F_s$

	Feature Space		
	$2N$-D Original	3-D Canonical	$2N$-D Canonical
Test Escape Identification rate	12.5%	19.7%	20.3%
Application Runtime	176.19 s	4.63 s	19.44 s

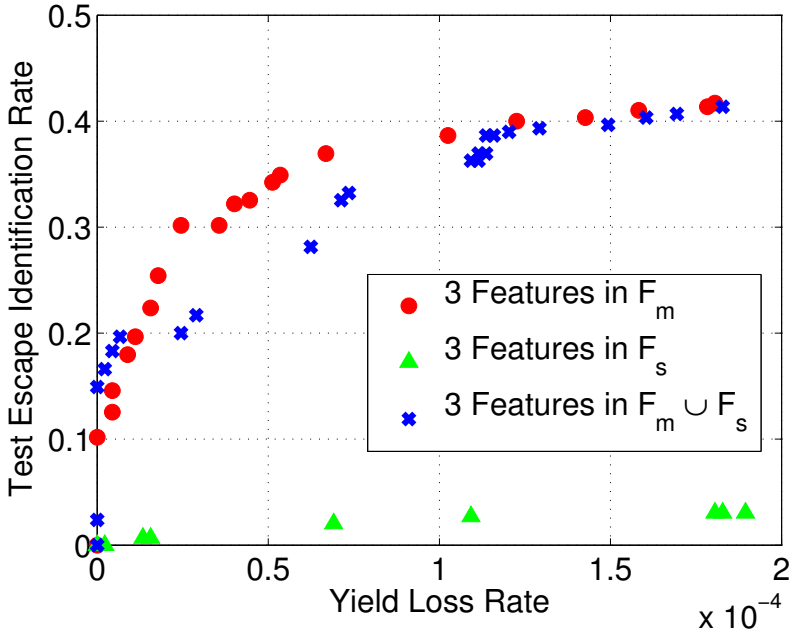


Figure 2.12: The ROC diagram for C-SVC in canonical space with 3 features derived from different feature sets, based on a data set with a reduced number of test escapes described in Section 2.5.5.

2.6 Summary

Through feature engineering, we propose two sets of features to characterize the health of chips. We demonstrate that statistical tests based on each set of

features could uniquely identify some test escapes that the other set of features cannot reveal. As adding more features may reveal more test escapes, we further propose to transform these features into a canonical space for feature reduction. Classification performed by the statistical tests on the reduced dimensions in canonical space achieves 29X runtime reduction while achieving a significantly higher accuracy than PCA in our experiment. We can expect further improvement if more types of features are added into the framework, followed by feature reduction through canonical analysis. Using our data set with emulated test escapes, we demonstrated that classification in a 3-dimensional canonical space can achieve greater accuracy than that in the original space with 200+ dimensions.

While C-SVC is used as the classification engine to evaluate various aspects of feature engineering proposed in the chapter, other classifiers can also be used. The scheme to utilize statistical tests containing the proposed features is flexible and can be easily extended to accommodate more types of statistical tests.

Chapter 3

AdaTest

3.1 Introduction

Statistical tests [1], which are statistical analyses following the physical test program, could help screen test escapes and outliers utilizing the hidden correlations in test data without additional physical measurements. In recent studies for outlier screening, O'Neill [5] and Sumikawa *et al.* [2] investigated the distribution of the population under test using principal component analysis (PCA) on correlated test items, and Krishnan and Kerkhoff [24] explored multiple Mahalanobis distances as the metric for screening. Chen *et al.* [3] also demonstrated various data mining techniques in predicting system-level test (SLT) failures.

Daasch *et al.* [25] proposed a concept of *residual* for outlier screening and demonstrated its applications considering the neighboring chips as the reference for deriving the residual, called *nearest neighbor residual* (NNR), in [26, 27, 28, 29]. In Chapter 2, we proposed a feature engineering framework which utilizes two

types of distinct feature sets and effectively reduces the number of features required for effective classification of test escapes. Each chip was characterized by a *residual vector*, which is a high-dimensional vector consisting of the difference between the measured value and an expected value of each test item. Selecting different expected values as the reference for producing the residual vector, which results in different input features for the classifier, reveals different subsets of the test escapes. While more types of features could potentially carry more useful information for classification, the feature dimension however could be too high, resulting in degradation in both runtime performance and accuracy for classification. Therefore, the framework further applies a canonical transform to extract and compact the most useful information from a large set of candidate features into a much smaller set for effective and efficient classification.

The above framework, however, requires all potentially useful features to be first generated from the data captured by the test program before performing the canonical transform. The runtime and space required for generating and transforming the features, which need to be done during test application, are already significant for today's products. They are expected to grow further for future products whose feature sets will continue to grow in size due to their greater complexity and even more stringent quality requirements. To address this problem, this chapter proposes a new statistical test framework that adopts some of the key ideas behind the popular Viola-Jones [30] framework, which was originally designed for real-time face recognition. Our proposed framework, named *AdaTest*, contains a cascade of AdaBoost [31] classifiers and only a small amount of features which are actually used in the classification need to be generated during

test application. This significantly reduces the runtime and memory space needed for processing the test data and thus enables real-time application of statistical tests, e.g. running the statistical tests on the automatic test equipment (ATE) during wafer probe tests for minimum adjustment of the production test flow and additional test time for high volume products. Moreover, we demonstrate that AdaTest and the method proposed in Chapter 2, which employs a support vector machine (SVM) as the classifier, could each identify some unique test escapes that cannot be identified by the other method. Therefore, a hybrid method combining the results of the two distinct classifiers could further improve the accuracy of test escape detection.

We introduce a new residual vector in addition to the two residual vectors used in Chapter 2 in this chapter. We demonstrate that including the proposed third type of residual vector as an input feature set could improve the detection rate for test escapes. The usefulness of each type of residual vectors for test escape detection is data-dependent (this will be demonstrated later in the experimental results section). Thus it should be a winning strategy to develop more sets of residual vectors that are potentially useful, and for each dataset/product, apply AdaTest to automatically select only the relevant features for classification of the target product.

The rest of the chapter is organized as the following. Section 3.2 illustrates the AdaTest framework, Section 3.3 discusses the data preparation and feature generation for the framework, and Section 3.4 demonstrates experimental results. Section **??** concludes the chapter.

3.2 AdaTest

In this section we illustrate the proposed statistical test framework, *AdaTest*, which consists of a cascade of AdaBoost classifiers. We first introduce the AdaBoost algorithm, followed by the algorithm for training the cascade of AdaBoost classifiers given a yield loss budget.

3.2.1 AdaBoost

Adaptive boosting, or *AdaBoost* [31], is a technique for combining multiple base classifiers to form one final classifier which can significantly outperform any of the base classifiers. The base classifiers are often referred to as *weak classifiers* and the final classifier is referred to as a *strong classifier*. The weak classifiers can be very simple and only need to be at least better than random classification, e.g., a classifier with an accuracy of 51% is acceptable as a weak classifier. The key idea is to train the weak classifiers iteratively, and in each iteration increase the weight of the misclassified samples in the cost function. Later weak classifiers will therefore focus more on the misclassified samples, and the final classification result is the weighted sum of the weak classifiers' classification results.

In Viola and Jones's framework for real-time face recognition [30], each weak classifier is simply a binary classifier based on one feature, referred to as a *decision stump*. Multiple decision stumps form a strong classifier, and multiple strong classifiers are iteratively trained and added to a cascade until a pre-set accuracy is reached. Given a set of samples x_i, $i \in 1...N$, whose class labels y_i are 1 for positives and -1 for negatives respectively, Algorithm 1 [32] trains a strong

classifier with T weak classifiers. For a statistical test whose objective is to screen test escapes, we define test escapes as positive samples and good chips as negative samples. Since in production test, the number of test escapes is significantly smaller than that of the good chips, we set the initial weighting coefficient of the positive samples r_w times of that of the negative samples for more effective classification. In our experiment, r_w is set to 2 empirically.

Algorithm 1 AdaBoost

1 Initialize weighting coefficients $w_{t,i} = \frac{r_w}{N}$, $\frac{1}{N}$ for $y_i = 1$, -1 respectively.

2 **for** $t = 1, ..., T$ **do**

3 Select a weak classifier $y_t(x)$ which minimizes the weighted error function.

$$E_t = \sum_{i}^{N} w_{t,i} \mathrm{I}(y_t(x_i) \neq y_i)$$

where $\mathrm{I}(y_t(x_i) \neq y_i) = 1$ when $y_t(x_i) \neq y_i$, and $\mathrm{I}(y_t(x_i) \neq y_i) = 0$ otherwise.

4 Evaluate the quantity

$$\epsilon_t = \frac{\sum_i^N w_{t,i} \mathrm{I}(y_t(x_i) \neq y_i)}{\sum_i^N w_{t,i}}$$

and

$$\alpha_t = \ln(\frac{1 - \epsilon_t}{\epsilon_t})$$

5 Update the weighting coefficients

$$w_{t+1,i} = w_{t,i} \exp\{\alpha_t \mathrm{I}(y_t(x_i) \neq y_i)\}$$

6 The final strong classifier is given by

$$Y_T(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t y_t(x)\right)$$

In each of the T iterations, a feature and its corresponding threshold which

minimizes the cost function are selected as a weak classifier in step 3. In step 4 the normalized classification error ϵ_t of the current weak classifier is calculated and a coefficient α_t, which is a function of ϵ_t, is derived for updating the weights of the misclassified samples in step 5. α_t is also used as the weight for the tth weak classifier's classification result in deriving the final classification result in step 6. When $\epsilon_t = 0.5$, α_t equals to 0, which leads to no contribution from the tth classifier for the final decision, since the classifier's accuracy is the same as random guessing. The value of α_t increases as ϵ_t decreases, so that the classification results of the more accurate weak classifiers contribute more to the final decision, and as long as the prediction of the tth weak classifier is better than random guessing, its weight α_t will stay positive.

Using decision stumps as the weak classifiers, step 3 selects one feature and a threshold in each iteration, which provides comprehensible diagnostic information about the test escapes. By investigating the features selected in the classifier and the their corresponding weighting coefficients in the final strong classifier, one can learn the features based on which the test escapes are distinguishable and the relative significance of the features in separating the test escapes from the good chips.

3.2.2 Cascaded AdaBoost Classifiers

To screen test escapes, we build a cascade of AdaBoost classifiers using decision stumps as the weak classifiers. The chips that pass the physical test program go through this cascade of statistical tests for further screening for test escapes, as shown Fig. 3.1.

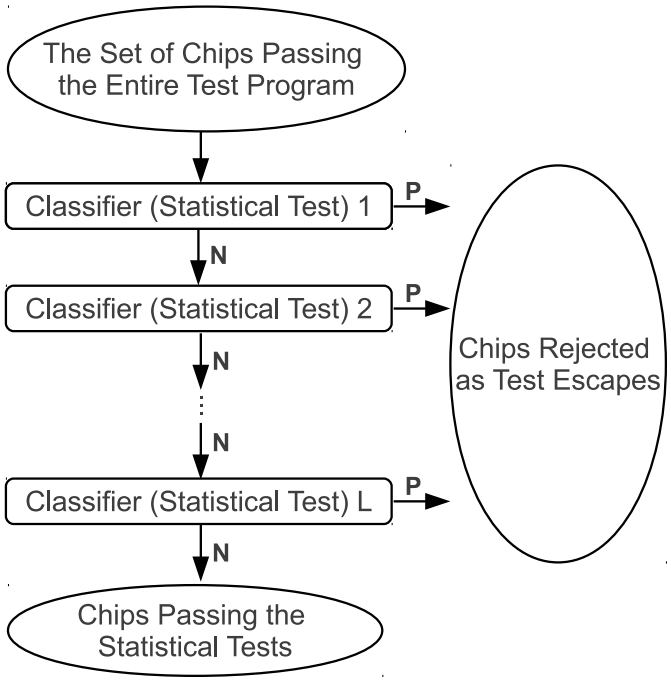


Figure 3.1: The cascade of classifiers for test escape screening which is applied after the physical test program for all chips in a wafer is completed.

Algorithm 2 illustrates the training process for designing the cascade to maximize the detection rate within a user-specified yield loss budget Y_{budget}. The input to the algorithm is a set of training samples consisting of known positives (test escapes) and negatives (good chips). The cascade contains multiple AdaBoost classifiers, each considered as a layer of the cascade. Given a yield loss limit per layer y_{layer}, the algorithm trains one AdaBoost classifier with a yield loss rate y lower than y_{layer} per iteration. The iterative process continues until Y_{budget} is reached or when a new layer of the cascade could not identify any new test escape within j_{max} weak classifiers, i.e., when the detection rate of test escapes d of the layer reaches zero. During the training phase, correctly identified test escapes in one layer will be excluded from the training set for training the next layer, so that the next AdaBoost classifier can focus only on the unidentified test escapes. All good chips, either correctly classified or misclassified at each layer, will remain in the training set for all layers of the cascade.

3.3 Data Preparation and Feature Generation

3.3.1 Data Standardization

There exist wafer-to-wafer and lot-to-lot variations in production test data, which deteriorate the robustness of applying a learned model based on a set of training wafers to query wafers. To address this problem, we standardize the test data of each wafer before further analysis. For each test item in every wafer, we first identify chips with outlying measurements, and derive the *robust mean*

Algorithm 2 Training the Cascaded AdaBoost Classifiers

```
1  S_p = set of positive training samples (i.e. test escapes)
2  S_n = set of negative training samples (i.e. good chips)
3  i = 0
4  while Y_i < Y_budget do
5      i = i + 1
6      j = 0
7      while y > y_layer do
8          j = j + 1
9          Train an AdaBoost classifier with j weak classifiers based on S_p and S_n
10         Evaluate the yield loss rate y and the detection rate d for the current
           layer
11         if d == 0 and j > j_max then break
12     Evaluate the overall yield loss rate Y_i based on the current cascade of
       classifiers
13     Exclude correctly identified test escapes from S_p
14     if d == 0 then break
```

and *robust standard deviation* which are the mean and standard deviation of the population excluding the outliers. Then the measurements in each wafer are standardized to *z-score*, with the robust mean and standard deviation by:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

where x is the measurement value, and μ and σ are the robust mean and standard deviation of the measurements for chips on the same wafer, respectively.

For identifying the outliers in each wafer, we use the general Extreme Studentized Deviate (ESD) test [33]. Given an upper bound for the number of outliers h, the general ESD test essentially performs h hypothesis tests: a test for one outlier, a test for two outliers, and so on up to h outliers to conclude the number of outliers and identify them. Details on the general ESD test is out of the scope of this chapter and can be found in [34]. Since the outliers in each wafer are likely to be caused by various random effects, the bias in the mean and standard deviation of each wafer is unpredictable and should be viewed as noise. Therefore it is necessary to exclude the outliers before deriving the statistics for standardizing the measurements on each wafer. After such standardization, the bias in the measurement data caused by wafer-to-wafer variations is reduced.

3.3.2 Features for Classification

In this study we characterize the chips using the *residual vectors* proposed in Chapter 2. Each chip with M test measurements is characterized by an $M \times 1$

residual vector $\boldsymbol{r}$:

$$\boldsymbol{r} = \boldsymbol{x_m} - \boldsymbol{x_e} \tag{3.2}$$

where $\boldsymbol{x_m}$ is an $M \times 1$ vector of the measured values and $\boldsymbol{x_e}$ is an $M \times 1$ vector of the expected values.

A residual vector represents how the measurement values of a chip deviate from its expected values, and is used as the set of input features for classification. In Chapter 2, the mean of measurements in the wafer and the spatial pattern learned through bilateral filtering [18] were used as two possible expected values for generating the residual vectors and thus produced two distinct feature sets for identifying test escapes. Note that other techniques such as virtual probe [8, 15] and Gaussian process model [9] could also be used for deriving the spatial patterns, but we apply bilateral filters in this study for its simplicity and speed. The residual vectors represent each chip's multi-dimensional deviation to the mean of the population in the wafer and to the systematic spatial variation of the wafer. It was reported that each of these two feature sets could reveal a unique subset of the test escapes.

There have been studies showing that comparing a chip's measurements with that of its neighbors could be used to reveal abnormalities of the query chip [10, 25, 35]. For test escapes that are defective chips, it is likely that there exists some difference between the test escapes and their neighbors. Therefore, in this study we propose a third feature set, for which the expected value used for producing the residual vector is the median of the target chip's eight surrounding neighbors' measurement values. Fig. 3.2 illustrates the location of the neighboring chips for a target chip marked t in the middle. This new feature set can be considered as

a special case of NNR [25]. Section 3.4 will show experimental results illustrating that the third feature set does provide additional information beyond the first two feature sets and improves the classification accuracy for the dataset analyzed in this study. Note that in this study, the selection of the eight nearest neighbors for reference is a general strategy that is likely to work on multiple products. The optimal selection of the most informative neighbors is product-specific and discussions on finding the optimal set of neighbors can be found in [26].

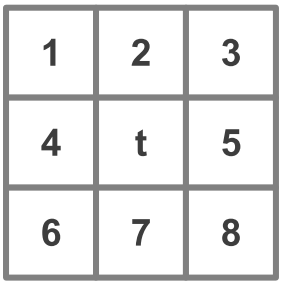


Figure 3.2: The median among the measurements of eight neighbors is used as the expected value for the target chip t in the middle.

3.4 Experimental Result

In this section we present the results of applying the proposed framework on a continue-on-fail production test data of an industrial product. The test data was preprocessed to remove confidential information while accurately preserving all information relevant to the analysis. The dataset consists of more than 700 wafers and we partitioned them into two groups: 200+ wafers as the training set and 500+ wafers as the testing set. Each wafer consists of 1000+ chips and the test program has more than 200 parametric test items.

Since the actual test escape information is not available in this dataset, we first introduce how we *emulated* test escapes and then demonstrate the experimental setup and results.

3.4.1 Emulating Test Escapes

Similar to the setup proposed in Chapter 2, the idea is to emulate test escapes using intrinsically defective chips which have subtle syndromes in their test measurements. To create a scenario in which the emulated test escape population has sufficient diversity and the test escape rate falls in a range of practical interest, after generating an initial pool of emulated test escapes based on the process described above, we further removed a fraction of them to form the final pool. The goal is to produce an emulated test escape pool such that the corresponding test items of those hidden failing measurements are widely spread among a large number of test items and no single test item is responsible for too many escapes (otherwise the pool would not be sufficiently diverse). Therefore, in the initial pool, we identified those test items that hiding each of them would contribute greater than 50PPM to test escapes, and removed those test escapes created by hiding these test items from the pool. The resulting test escape pool after this post-process corresponds to approximately 560PPM for the testing set.

3.4.2 Classification Accuracy

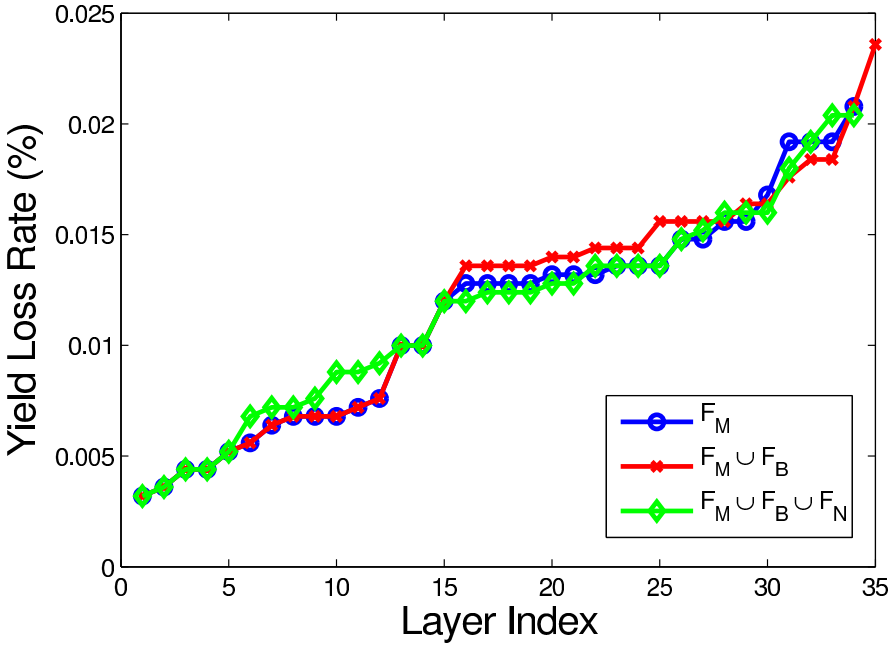
For simplicity and clarity, we use the following notations for the three feature sets we derived in Section 3.3.2:

- Feature Set F_M: Residual vectors derived using the mean of the measurements in the wafer as the expected values
- Feature Set F_B: Residual vectors derived using the learned spatial patterns via Bilateral Filtering as the expected values
- Feature Set F_N: Residual vectors derived using the median of the eight neighboring chips' measurements as the expected values

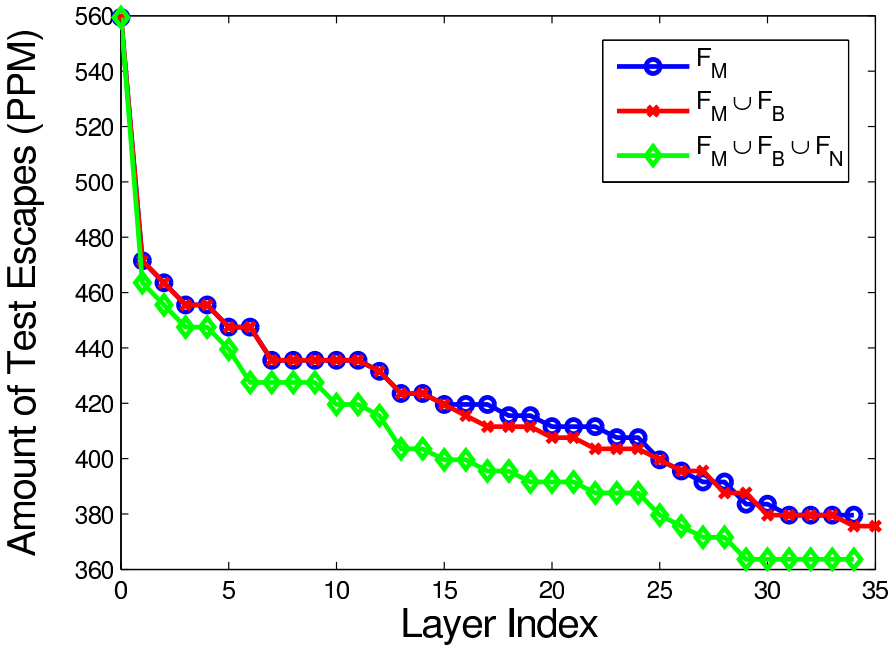
Fig. 3.3 demonstrates the classification accuracy of the cascaded classifiers on the testing set, with y_{layer} set to 0.01%, based on three choices of the input features for classification. Let M be the number of test items, the three choices are:

- F_M: Use the M features in F_M alone as the input features
- $F_M \cup F_B$: Use the $2M$ features in F_M and F_B jointly as the input features
- $F_M \cup F_B \cup F_N$: Use the $3M$ features in all three feature sets as the input features

Fig. 3.3a shows that the accumulated yield loss rate increases as more layers of AdaBoost classifiers are added to the cascade, and Fig. 3.3b shows that the test escape rate drops from 560PPM to 360PPM when all three feature sets are used as the input features. Although in Chapter 2, using $F_M \cup F_B$ as the input features resulted in significantly higher classification accuracy than using F_M alone, including F_B as the input features does not help detect more test escapes for the dataset in this study. However, it is clear from Fig. 3.3b that including the third feature set F_N as input to the classifiers could reveal more test escapes. Since AdaTest could automatically select the most useful features, we can still keep F_B



(a) The accumulated yield loss rate versus the number of layers



(b) The remaining amount of undetected test escapes versus the number of layers

Figure 3.3: The accumulated yield loss rate and the amount of remaining undetected test escapes versus the number of layers in the cascade of classifiers.

in a general collection of potentially useful feature sets as long as the feature set reveals some test escapes in some datasets. We can then apply such collection of feature sets to all new datasets/products and let AdaTest select the most useful features for each dataset/product.

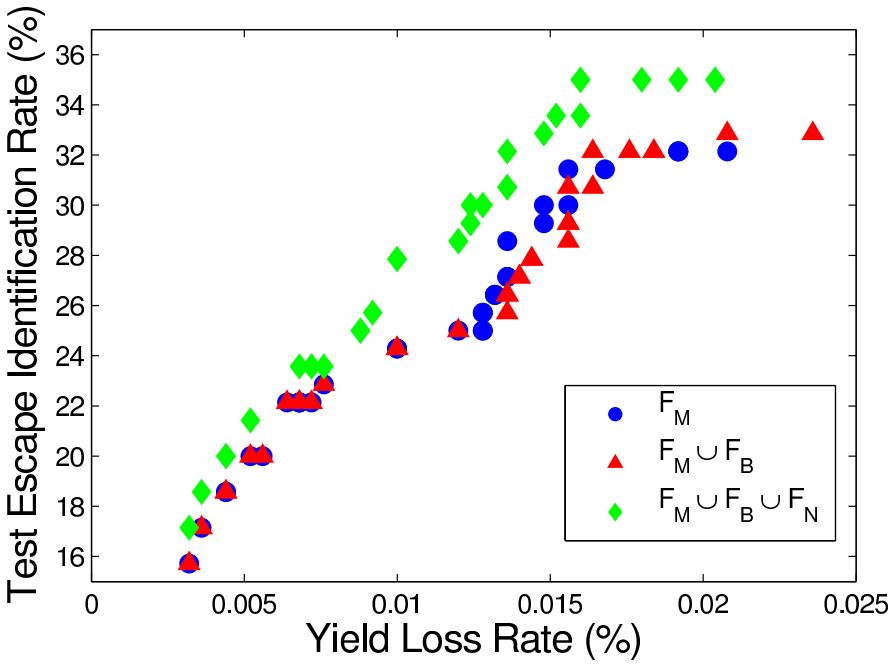


Figure 3.4: The ROC curves of classification based on different choices of input features.

Fig. 3.4 plots the relative operating characteristics (ROC) curves of classification, i.e. the test escape identification rate vs. the yield loss rate, based on different choices of input features. Given a yield loss rate budget, say 0.01%, classification based on $F_M \cup F_B \cup F_N$ identifies additional 4% out of the test escape pool, in comparison with those based on F_M and $F_M \cup F_B$. The results based on F_M and $F_M \cup F_B$ are similar for this dataset, and because the data characteristics of the training and the testing sets may exist slight differences, the classification

performance for the testing set based on F_M could sometimes slightly surpass that based on $F_M \cup F_B$ at certain yield loss rates.

The experimental results in Chapter 2 demonstrated significant classification accuracy improvement compared with PCA. Fig. 3.5 shows the accuracy comparison between AdaTest and the SVM-based framework in Chapter 2, in which the first three dimensions in the canonical space were used as the input features for SVM. To make a fair comparison, we use $F_M \cup F_B \cup F_N$ as the input features for both frameworks. The ROC curves of the SVM-based framework and AdaTest show that the former achieves a higher test escape detection rate at a given yield loss rate. However, further investigation of the specific test escapes identified by each of the two frameworks shows that the two approaches identify different subsets of the test escapes.

Fig. 3.6 shows the Venn diagrams of the identified test escapes among the entire test escape population and the yield loss populations for the two frameworks, at a yield loss rate of 0.01%. In Fig. 3.6a, while the SVM-based framework and AdaTest identify 30% and 27.9% of the test escapes respectively, there are only 18.6% out of all the test escapes that are identified by both frameworks. Each of the two frameworks uniquely identifies 11.4% and 9.3% test escapes. In Fig. 3.6b, out of the 0.01% yield loss, each framework causes 0.0084% unique yield loss. Based on these results, a hybrid framework incorporating both AdaTest and the SVM-based method could likely achieve better performance than each individual method alone. Just using the most naive idea which takes the union of the results based on these two methods could result in a yield loss rate of 0.0184% and a test escape detection rate of 39.3%, which outperforms the two methods by

at least 3.3% at the corresponding yield loss rate, as shown in Fig. 3.5. The implementation of a tightly integrated hybrid framework which could optimally incorporate AdaTest and the SVM-based method is currently under development.

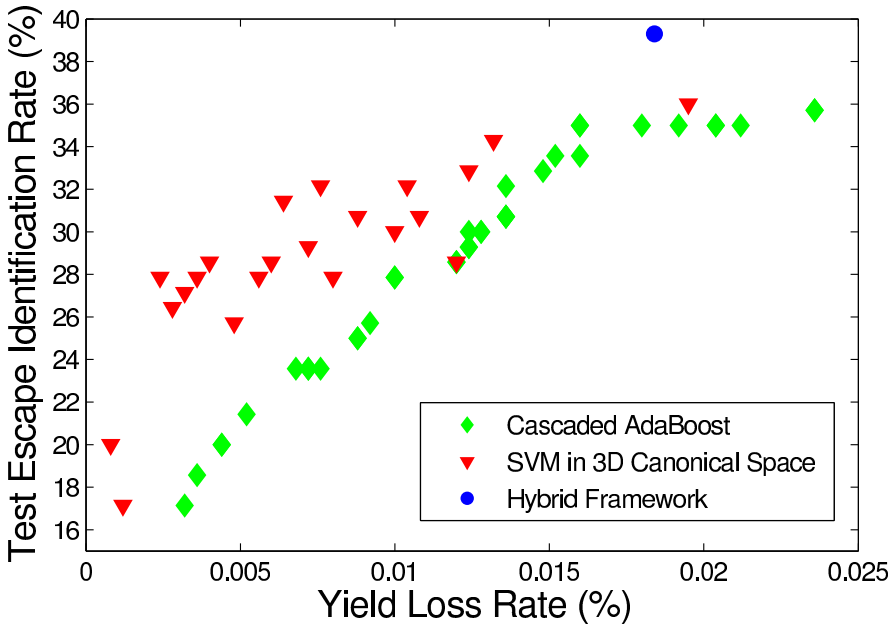
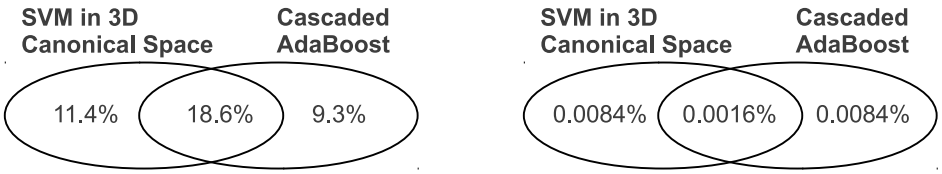


Figure 3.5: The ROC curves of classifications based on cascaded AdaBoost and SVM in 3-dimensional canonical space.



(a) Identified test escapes (b) Yield loss

Figure 3.6: The Venn diagram of the populations of identified test escapes and yield loss for the SVM-based framework and AdaTest.

Another observation from our experiment is that with the yield loss limit per layer y_{layer} set at 0.01%, all layers of AdaBoost classifiers contain only one weak classifier per layer. That is, the entire framework is composed of a cascade of

decision stumps, each of which selects one feature and a corresponding threshold for binary classification. During the training process, we have explored various settings which might enable the AdaBoost classifiers to train multiple weak classifiers per layer. While some of the resulting classifiers did improve the test escape detection rate for the training set, the improvement in training was not necessarily observed in the testing set. This result implies that the training of incorporating more weak classifiers causes overfitting in the training set and fails to generalize to other sample sets.

While the above observation may be a unique result of the specific dataset or a more general phenomenon for test data, we can learn the following from the results of this data: To use the AdaBoost algorithm with decision stumps for our application, training a more complicated strong classifier consisting of multiple weak classifiers in one layer is less effective than training more layers of very simple strong classifiers, each of which consists of only one or few weak classifiers. Each of these strong classifiers targets a unique and small subset of test escapes in one layer. As the characteristics of the test escapes could be very diverse, each small cluster of the escapes could be identified by a simple classifier. Combining multiple simple classifiers into a single, more complex classifier may lose their unique individual strengths for detecting small clusters in a diverse population.

3.4.3 Application Runtime and Memory Usage

While more distinct features may potentially improve classification accuracy for test escapes, developing a large, generic collection of potentially useful feature sets across products increases the runtime and memory usage during test applica-

tion and could limit the real-time application of statistical tests. Table 3.1 shows the runtime for generating each of the three feature sets in this study for each wafer which has 200+ parametric test items and 1000+ chips, in an Intel Xeon Quad-core 3.6GHz system. In the SVM-based framework, all feature sets need to be generated from the test data, which takes 1.339 seconds per wafer. At a yield loss rate of 0.01% and a test escape detection rate of 27.9%, AdaTest selects 7 features from F_M, no feature from F_B, and 7 features from F_N, which takes 0.016 seconds per wafer for generating the feature sets. The runtime of each step in the two frameworks is listed in Table 3.2. Since AdaTest generates only the useful features from test data, it does not require a feature transformation phase. On average, AdaTest achieves 83X runtime reduction during test application. In the naive hybrid framework mentioned in Section 3.4.2, at a yield loss rate of 0.0184%, we could increase the runtime from 1.489 seconds to 1.507 seconds (a 1.2% runtime increase) for an additional 3% test escape detection beyond the SVM-based method.

Table 3.1: Runtime Per Wafer for Generating Each Feature Set From the Test Data

	Feature Set		
	F_M	F_B	F_N
Runtime (s)	0.027	0.768	0.544

The runtime for generating and transforming the features for F_M, $F_M \cup F_B$, and $F_M \cup F_B \cup F_N$ is shown in Fig. 3.7. The runtime for preparing the features in the SVM-based framework, in which all potentially useful features need to be generated, becomes significantly greater than that of AdaTest as the size of the

Table 3.2: Runtime Per Wafer for Each Step in the Statistical Test Frameworks Given $F_M \cup F_B \cup F_N$ as Input Features

	Feature Set	
	SVM-based framework	AdaTest
Feature Generation (s)	1.339	0.016
Feature Transformation (s)	0.037	-
Classifier Application (s)	0.113	0.002
Total (s)	1.489	0.018

input feature sets grows. Note that AdaTest selects exactly the same features from F_M when given F_M and $F_M \cup F_B$ as the input features to reach a yield loss rate of 0.01%, which is shown in Fig. 3.4. Therefore the runtime for preparing the features does not change when including F_B in addition to F_M as input features.

During test application, the memory space needed for storing the 3 types of features, F_M, F_B, and F_N, before the canonical transform is at least 3X of the original test data. In a naive implementation of the transform, a huge N by $3M$matrix containing all the feature information is multiplied by another transform matrix of size $3M$ by $3M$, where N is the number of samples and M is the number of test items where in our experiment $M > 200$. On the other hand, AdaTest identifies 14 features in the training phase and uses them directly, without any further transformation, for classification in the test application phase. Therefore, AdaTest consumes significantly less memory for processing the features than the SVM-based method.

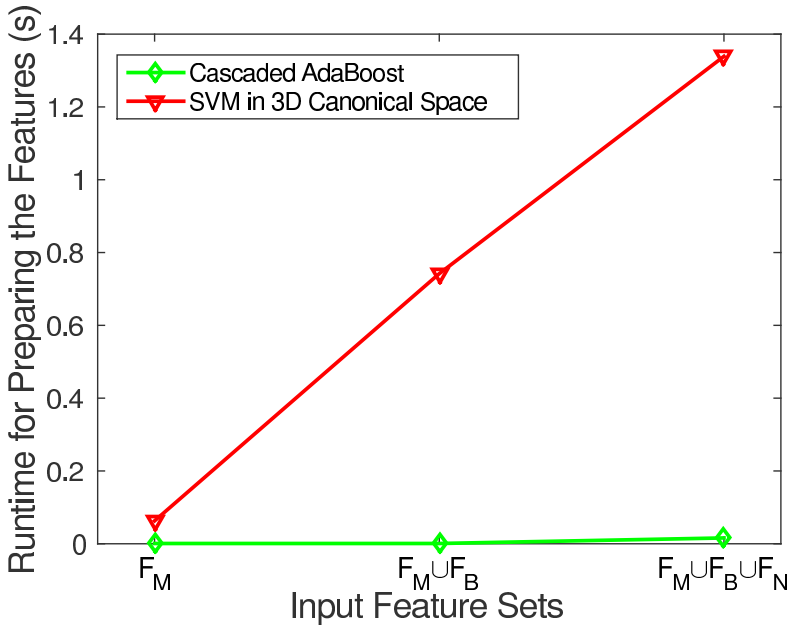


Figure 3.7: The runtime for generating and transforming the features before classification.

3.4.4 Feature Selection

As mentioned in Section 3.2.1, another advantage of AdaTest is that it selects features directly without transformation, which provides comprehensible diagnostic information about the test escapes. In this study, the first three layers of the cascade selected a feature in F_N calculated based on a standby current measurement, a feature in F_N based on a digital-to-analog converter (DAC) performance measurement, and a feature in F_M based on a standby current measurement. Out of the 14 selected features given a yield loss rate of 0.01%, no feature from F_B was selected. The majority of the selected features include features in F_M and F_N based on standby current measurements, DAC performance measurements, and analog-to-digital converter (ADC) performance measurements. Although the de-

tails of the test items could not be revealed, the above example demonstrates the information this framework provides for better understanding the characteristics of the test escape population.

3.5 Summary

In this study, we propose a framework, *AdaTest*, for designing statistical tests which consists of a cascade of AdaBoost classifiers using decision stumps as the weak classifiers. Given a collection of potentially useful feature sets, AdaTest can identify a small number of features in the training phase that are most useful for classification. In contrast, an SVM-based method, which can also achieve high classification accuracy, needs to produce the entire collection of feature sets, followed by a canonical transform for feature reduction before performing classification. Therefore, AdaTest significantly reduces the runtime by *83*X and memory usage by at least *3*X compared with an SVM-based framework. Such improvement enables real-time application that can be carried out on the ATEs for high volume products, which minimizes the adjustment of the production test flow in test phases such as the wafer probe test.

We also demonstrate that a new feature set F_N, defined as the residual vector with respect to the median of eight neighbors' measurement values of the sample chip, could reveal more test escapes. Since AdaTest could automatically select the most relevant features, we can apply a general collection of potentially useful feature sets to a new dataset, and the unhelpful features for the specific dataset will be automatically excluded, such as F_B in this study.

As AdaTest and the SVM-based method each identifies a unique subset of test escapes, a hybrid framework integrating these two methods and combining their strengths could further improve the detection rate of test escapes without taking any additional physical test measurements.

Chapter 4

Proximity-Based Features

4.1 Introduction

In this chapter, we propose new features that are based on the pairwise proximities calculated from the abovementioned three feature sets, which are referred to as the *base feature sets* in the rest of the paper. Given a test data with T test items and D chips, we calculate a $D \times D$ proximity matrix based on a selected proximity/distance function in the $3T$ feature space constructed from the base feature sets. Different distance functions could potentially provide unique information that reveals the abnormalities of some test escapes. We investigate six different distance functions including *cosine distance*, *correlation distance*, and *Minkowski distance* with $p = 1, 2, 3, \infty$ for deriving the proximities.

The *proximity representation* of a dataset (represented by a $D \times D$ proximity matrix), however, could not be analyzed with the traditional machine learning algorithms that are designed for a Euclidean space, or a *vector representation*

(represented by a $D \times T$ matrix containing all the feature values for all samples). Therefore, we apply a technique named *constant shift embedding* (CSE) [36] to convert the proximity information back into a Euclidean space. The most prominent property of CSE is the complete preservation of cluster structure in the embedded Euclidean space. The six distance functions would lead to six unique proximity matrices, and therefore results in six unique embedded spaces. In addition, we further investigate a traditional kernel PCA (kPCA) embedding method [37] with radial basis function (RBF) kernel and generate a seventh Euclidean space to provide even more information that could potentially separate test escapes.

In each of the seven unique embedded spaces constructed based on the pairwise proximities calculated from the base feature sets, we apply a density-based outlier analysis called *local outlier factor* (LOF) [38]. LOF compares a sample's local density with its neighbors' densities in a feature space and produces a single value. A sample with a relatively higher LOF value than that of the majority of the samples is more likely to be an outlier. We also observed that in the first dimension of each embedded space, some test escapes are away from the good chip population, which makes them easily separable. Therefore, we use the LOF value and the first dimension of each of the embedded spaces jointly as the new proximity-based features. Given the seven proximity definitions, 14 new features are generated. Based on these new features plus the $3T$ features from the base feature sets, we then perform feature reduction using the canonical analysis proposed in Chapter 2. Canonical analysis is a linear transformation that maximizes the separation between the two populations of samples (good chips and test escapes) in the first few dimensions of the transformed feature space,

called a canonical space. A classical classifier such as the support vector machine (SVM) [39, 22] can then be applied in the canonical space for more efficient and in some cases, more effective classification. With these proximity-based features which provide additional revealing information about test escapes, the test escape detection rate based on an industrial production test dataset is improved to 31%, compared with 27% for similar analysis using the base feature sets only.

The rest of the chapter is organized as the following: Section 4.2 introduces the distance functions used for generating the pairwise proximities between samples. Section 4.3 illustrates the concepts and properties of constant shift embedding, followed by a discussion about the distribution of the chips in the embedded space. Section 4.4 discusses data standardization, feature generation, outlying wafer detection, and feature transformation using canonical analysis. Additional experimental results are presented in Section 4.5, and Section **??** concludes the chapter.

4.2 Pairwise Proximity

In this chapter we extract more information to reveal the abnormalities of the test escapes by comparing each chip with all the other chips on the same wafer. The comparison is made in a feature space composed of the three base feature sets developed in Chapter 3, in which each chip with T test measurements is characterized by a $T \times 1$ residual vector $\boldsymbol{r}$:

$$\boldsymbol{r} = \boldsymbol{x_m} - \boldsymbol{x_e} \tag{4.1}$$

where $\boldsymbol{x_m}$ is a $T \times 1$ vector of the measured values and $\boldsymbol{x_e}$ is a $T \times 1$ vector of the expected values. Three expected values were used to produce three base feature sets: the mean of the measurements on the same wafer, a value based on the bilateral filtered [18] spatial pattern of the wafer, and the median of the eight nearest neighbors of the query chip. The third feature set can be considered as a special case of NNR [25]. We denote the three generated feature sets as F_M, F_B, and F_N respectively. Throughout the analysis in this chapter, we will be using the three feature sets jointly, denoted $F_M \cup F_B \cup F_N$, as our base feature space for deriving the proximities.

Given a wafer with D chips, the pairwise comparisons between each pair of chips result in a $D \times D$ symmetric proximity matrix, each of whose elements represents the pairwise proximity between two chips. Our strategy is to generate all potentially useful features, followed by a feature reduction technique such as canonical analysis to automatically extract the most useful information out of the large set of generated features for classification. Therefore, we apply multiple different distance functions for calculating the pairwise proximity between each two chips to potentially reveal more aspects of the abnormalities of test escapes with the conjecture that each of these distance functions might uniquely separate some of the test escapes from the normal populations. Let $\boldsymbol{x_a}$ be a $T \times 1$ vector consisting of sample a's feature values $x_{a1}, x_{a2}, ..., x_{aT}$, the distance functions we investigated are:

- *Cosine distance*:

$$d_{ab} = 1 - \frac{\boldsymbol{x'_a x_b}}{\sqrt{(\boldsymbol{x'_a x_a})(\boldsymbol{x'_b x_b})}} \tag{4.2}$$

- *Correlation distance*:

$$d_{ab} = 1 - \frac{(\boldsymbol{x_a} - \overline{\boldsymbol{x}}_{\boldsymbol{a}})'(\boldsymbol{x_b} - \overline{\boldsymbol{x}}_{\boldsymbol{b}})}{\sqrt{(\boldsymbol{x_a} - \overline{\boldsymbol{x}}_{\boldsymbol{a}})'(\boldsymbol{x_a} - \overline{\boldsymbol{x}}_{\boldsymbol{a}})(\boldsymbol{x_b} - \overline{\boldsymbol{x}}_{\boldsymbol{b}})'(\boldsymbol{x_b} - \overline{\boldsymbol{x}}_{\boldsymbol{b}})}} \tag{4.3}$$

  where $\overline{\boldsymbol{x}}_{\boldsymbol{a}}$ is the mean of vector $\boldsymbol{x_a}$.

The following four distances are derived from *Minkowski distance* with different values for parameter p:

$$d_{ab} = \sqrt[p]{\sum_{j=1}^{T} |\boldsymbol{x_{aj}} - \boldsymbol{x_{bj}}|^p} \tag{4.4}$$

- $p = 1$ *(Manhattan distance)*:

$$d_{ab} = \sum_{j=1}^{T} |\boldsymbol{x_{aj}} - \boldsymbol{x_{bj}}| \tag{4.5}$$

- $p = 2$ *(Euclidean distance)*:

$$d_{ab} = \sqrt{(\boldsymbol{x_a} - \boldsymbol{x_b})'(\boldsymbol{x_a} - \boldsymbol{x_b})} \tag{4.6}$$

- $p = 3$:

$$d_{ab} = \sqrt[3]{\sum_{j=1}^{T} |\boldsymbol{x_{aj}} - \boldsymbol{x_{bj}}|^3} \tag{4.7}$$

- $p = \infty$ *(Chebyshev distance)*:

$$d_{ab} = \max_j |\boldsymbol{x_{aj}} - \boldsymbol{x_{bj}}| \tag{4.8}$$

In addition to the above six distance functions, we also include a traditional

kernel PCA method [37] with a radial basis function (RBF) kernel:

- *RBF/Gaussian kernel*:

$$k_{ab} = \exp\left(-\frac{(\boldsymbol{x_a} - \boldsymbol{x_b})'(\boldsymbol{x_a} - \boldsymbol{x_b})}{2\sigma^2}\right) \tag{4.9}$$

Each of the first six distance functions would lead to a unique proximity matrix, which will be further converted to a Euclidean space by CSE. For the proximity matrix generated using the RBF kernel, we apply the traditional kernel PCA algorithm for producing an embedded space without CSE to validate if the existing kPCA technique could also provide additional information.

4.3 Constant Shift Embedding

4.3.1 Concepts and Properties

After generating the proximity matrices based on multiple distance functions, we need to convert the proximity representation back into a vector representation before applying traditional outlier detection algorithms that are designed for a Euclidean vector space. *Constant shift embedding* (CSE) [36] is a technique to embed pairwise proximity data into the equivalent Euclidean embedding with no distortions. Specifically, CSE finds a Euclidean space in which the cost function of a Euclidean distance-based clustering algorithm such as *k-means* could be equivalent to the cost function of pairwise clustering on the proximity matrix. Detailed computations of CSE can be found in [36].

Fig. 4.1 shows the process of producing new embedded feature spaces based

on proximity matrices. From the original space O_1, which is composed of the base feature sets $F_M \cup F_B \cup F_N$, we generate multiple proximity matrices based on different distance functions, followed by CSE for each proximity matrix to convert them into embedded Euclidean spaces. While CSE preserves the cluster structure through the conversion from a proximity representation to a Euclidean vector representation (e.g., the cluster structure is preserved between E_1 and P_1, between E_2 and P_2, and so on), the k-means cost function in the original feature space O_1 would also be identical to the cost function of pairwise clustering in the proximity matrix derived using Euclidean distance P_1 [40]. Therefore, applying k-means clustering in E_1 is equivalent to applying k-means clustering in O_1. In this special case, in fact, the original space is already Euclidean. The added value of CSE in our analysis comes from the ability to assimilate also other arbitrary measures of proximity into more informative Euclidean spaces.

For a proximity matrix derived from O_1 with distance functions other than Euclidean distance, e.g. P_2 using cosine distance, we can also consider it is derived from a virtually equivalent original feature space O_2 using Euclidean distance. In such case, applying k-means in E_2 would be identical to applying k-means in O_2. However, with the use of nonlinear distance functions to derive the proximities, a direct transformation from O_1 to O_i, where $i \neq 1$, is often not feasible. Analyzing E_i through the calculation of P_i followed by CSE, achieves the same goal without the need of finding O_i.

CSE involves eigendecomposition of the proximity matrix [36]. That is, the embedded space is composed of the eigenvectors of the proximity matrix. In our application, we analyze only the first few dimensions, which have relatively

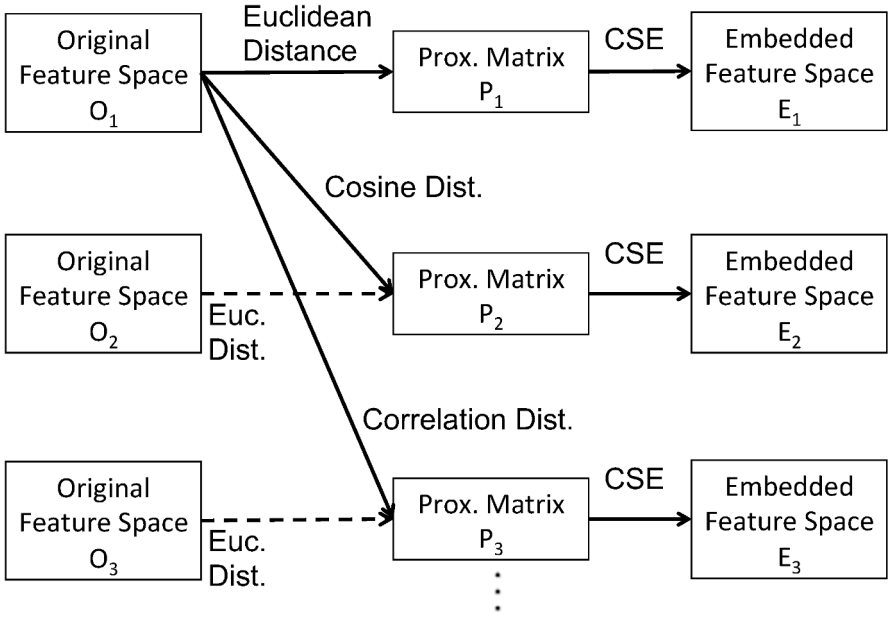


Figure 4.1: The conversion between proximity matrices and Euclidean spaces. CSE preserves the cluster structure through the conversion from a proximity matrix P_i to an embedded Euclidean space E_i.

significant eigenvalues, for feature reduction. Let u be the number of eigenvectors found from the eigendecomposition, and $ev_1, ev_2, ..., ev_u$ be the sorted eigenvalues such that $ev_1 \geq ev_2 \geq ... \geq ev_u$, we estimate the number of effective dimensions of the embedded space by:

$$D_{eff}(i) = \frac{\sum_{j=1}^{u} ev_j}{ev_1} \tag{4.10}$$

For each embedded space E_i, we apply the outlier analysis algorithm in its corresponding dimensionality of $ceil(D_{eff}(i))$.

Note that for the seventh Euclidean space, we apply the traditional kernel PCA approach with an RBF kernel for generating the proximity and deriving an embedded space, without applying CSE. Our overall strategy is to generate as many potentially useful features as possible. Since kPCA is known to be one of the potentially useful transformations, it is worthwhile to include it to enrich our analysis. In the experimental results demonstrated later, we validate that features based on E_i's and the kPCA space are both useful for further improvement of classification accuracy.

4.3.2 Distribution in the Embedded Space

For one exemplar wafer with two test escapes, Fig. 4.2 shows the distributions of good chips (blue dots) and the test escapes (red crosses) in the embedded spaces constructed based on the six types of proximities, with the number of effective dimensions marked above each distribution. The distribution in the embedded space constructed using kPCA is shown in Fig. 4.3.

It is clear that, in all the distributions, the good chip population exhibits a bimodal distribution - the good chips on a wafer are separated into two clusters through the proximity calculation and the CSE transformation. Fig. 4.4 shows the mapping of the distribution of good chips on the exemplar wafer in the embedded space constructed based on cosine distance to a wafer map. In Fig. 4.4a, chips are colored based on their locations, and the same colors are marked on the wafer map in Fig. 4.4b to indicate the corresponding locations of the chips on wafer. From the high-dimensional base feature space $F_M \cup F_B \cup F_N$, the proximities are able to reveal the underlying horizontal stripe pattern on the wafer even though in most test items this pattern are not directly observable and shadowed by some other more dominant types of spatial patterns. In other words, such stripe spatial variation may be subtle but consistently exists in most of the test items. Finding such hidden spatial patterns, which is feasible using the proposed analysis with proximities based on different nonlinear distance functions, could help the diagnosis of manufacturing/testing process and equipment such as multi-site probing.

Defined in (4.10), the number of effective dimensions in embedded spaces based on cosine, correlation, Manhattan (Minkowski with $p = 1$), and Euclidean (Minkowski with $p = 2$) distances are typically no greater than 3. In general, Minkowski distance with greater p leads to a greater number of effective dimensions, and Minkowski distance with a very small p, say 1, generates little information and is insufficient to expose the test escapes as outliers. Details of how to analyze the distributions for screening test escapes will be discussed in Section 4.4.2.

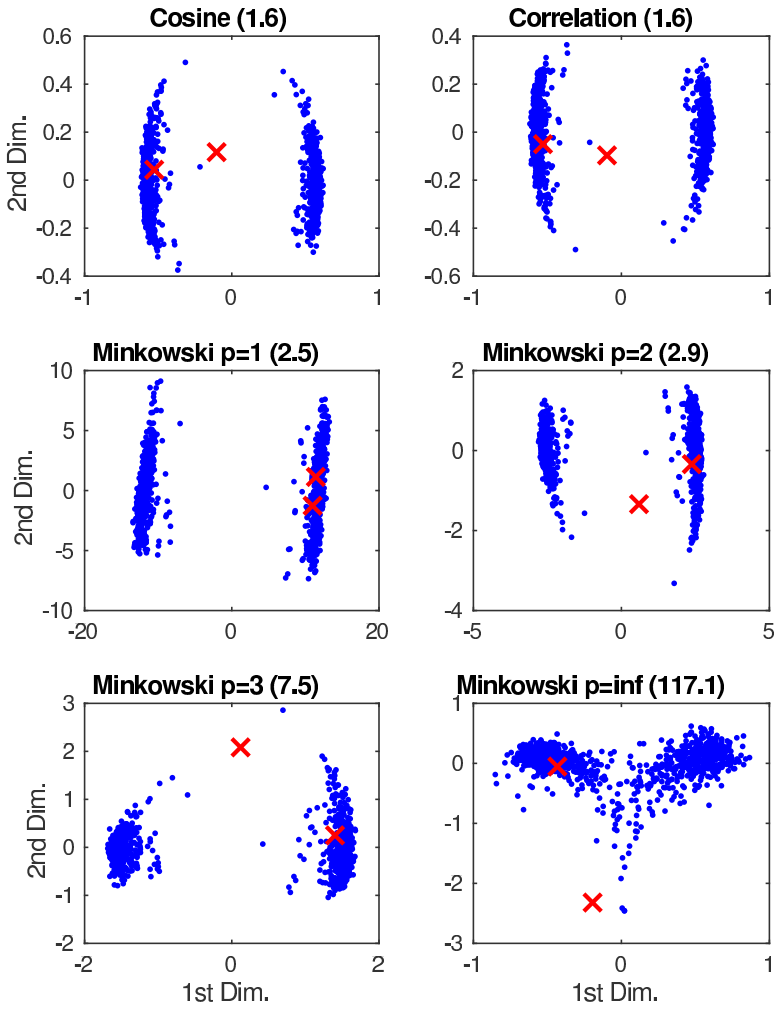


Figure 4.2: The distributions of the chips on a wafer in the first two dimensions of the CSE embedded spaces based on six different proximity/distance functions. Blue dots represent the good chips and red crosses mark the positions of test escapes. The numbers of effective dimensions are shown above each figure.

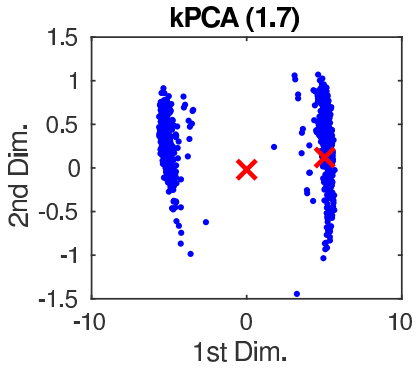
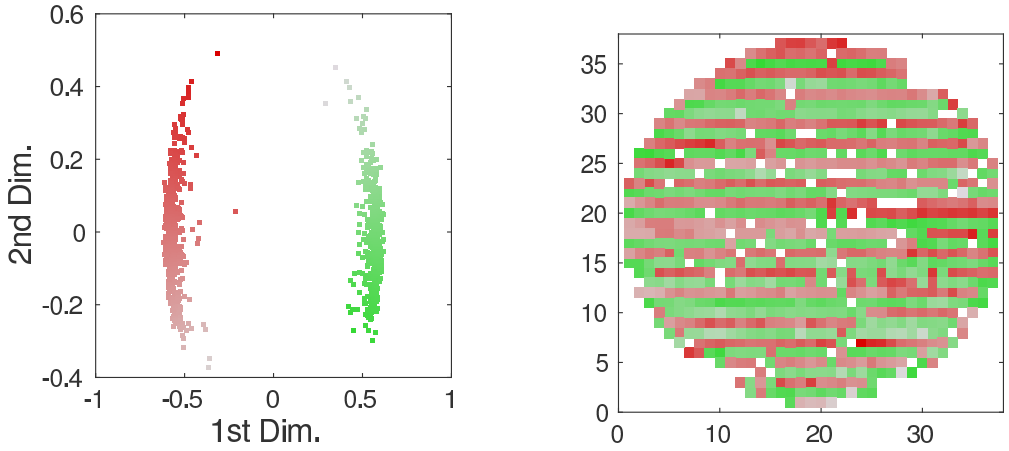


Figure 4.3: The distribution in the first two dimensions of the embedded space constructed based on kPCA with RBF kernel.



(a) Distribution of good chips in an embedded space

(b) Corresponding positions on wafer

Figure 4.4: Color-coded distributions of good chips showing the correspondence of chips in the embedded space and on the wafer. Chips are colored to show their corresponding positions.

4.4 Data Preparation and Feature Processing

In this section we discuss how we preprocess the production test data, generate new features from the embedded spaces, and transform the features for feature reduction. We also demonstrate a process to identify and remove some abnormal wafers from our analysis.

4.4.1 Data Standardization

As proposed in Chapter 3, to minimize the wafer-to-wafer variation in production test data, we first standardize the measurement values of each wafer before further analysis. For each test item in each wafer, we identify outlying measurements using the general Extreme Studentized Deviate (ESD) test [33], and calculate the *robust mean* μ and *robust standard deviation* σ excluding the outlying measurements. We then standardize the measurements x in each wafer individually to *z-score* by:

$$z = \frac{x - \mu}{\sigma} \tag{4.11}$$

Given an upper bound for the number of outliers h, the general ESD test performs h hypothesis tests: a test for one outlier, a test for two outliers, and so on up to h outliers to conclude the number of outliers and identify them. Detailed implementation of the general ESD test can be found in [34].

4.4.2 Feature Generation

As observed in Fig. 4.2, in all embedded spaces except one constructed based on Manhattan distance, which takes into account only the first order difference between chips, one of the two test escapes is exposed as abnormal and far from the bimodal distribution of the good chips, while the other test escape is indistinguishable from the good chips. Since our goal is to maximize the test escape detection rate while minimizing the amount of induced yield loss (good chips misclassified as test escapes), the classification accuracy would be higher if test escapes could be outlying in as many embedded spaces as possible, and the good chips that happen to be outlying in one embedded space to be closer to the normal population in other embedded spaces. Therefore, although one embedded space seems sufficient to expose the test escape as an outlier in Fig. 4.2, it improves the robustness of the method to include the distribution information in all embedded spaces for further analysis.

To analyze the distributions in multiple embedded spaces jointly, we convert the outlying level of each chip in each embedded space to a score, defined by *local outlier factor* (LOF) [38]. LOF is an outlier analysis algorithm that compares the local density of the sample with the densities of its neighbors. Let *k-distance*(p) be the distance between sample p and its k-th nearest neighbor, a *reachability distance* is defined by:

$$\textit{reach-dist}_k(p,q) = max\{\textit{k-distance}(q), d(p,q)\} \tag{4.12}$$

where $d(p,q)$ denotes the distance from p to q. Including *k-distance(p)* in the

reachability distance could produce a more stable result than using $d(p, q)$ directly.

Using a parameter $MinPts$ for k, the *local reachability density* of p is defined as:

$$lrd_{MinPts}(p) = 1/\left(\frac{\sum_{q \in N_{MinPts}(p)} reach\text{-}dist_{MinPts}(p, q)}{|N_{MinPts}(p)|}\right) \tag{4.13}$$

where $N_{MinPts}(p)$ is the set of $MinPts$ nearest samples of p. Discussions about choosing the upper and lower bounds for $MinPts$ can be found in [38]. In our analysis, the range is set to $5 \leq MinPts \leq 10$.

The local outlier factor is then defined as:

$$LOF_{MinPts}(p) = \frac{\sum_{q \in N_{MinPts}(p)} \frac{lrd_{MinPts(q)}}{lrd_{MinPts(p)}}}{|N_{MinPts}(p)|} \tag{4.14}$$

The LOF value is a relative value indicating the outlying level of a sample compared with its neighbors. Typically, an LOF value close to (greater than) 1 tends to indicate an inlier (outlier), but the actual threshold is data dependent. With the local density approach, a sample with some distance to a dense cluster could have a much greater LOF value than another sample with the same distance to a sparse cluster, and thus be exposed as an outlier.

Now that we can express the outlying level of each chip by a single LOF value, we use these LOF values as our new pairwise proximity-based features. Instead of setting a threshold directly on the LOF values, we use the LOF values jointly with other base features for machine learning algorithms such as SVM for classification. Another simple observation from the distributions is that the detectable test escapes, away from the bimodal distribution, are typically closer

to the origin in the first dimension of the embedded space. Therefore, we also include the first dimension of the embedded spaces as input features for further analysis. In total, 14 new features are generated from the 7 embedded spaces based on pairwise proximities.

4.4.3 Feature Standardization and Outlying Wafer Detection

There exist wafer-to-wafer variations in production test data, and we standardize each wafer individually with respect to the robust mean and standard deviation before any analysis to remove the shifting and scaling variations. However, although all the wafers we analyzed exhibit the bimodal distributions as in Fig. 4.2, we have observed noticeable variations in the 14 new features, especially the LOF values since they are relative values depending on the local distribution. Thus, we further standardize the new features generated from each wafer to z-scores using the robust mean and standard deviation calculated from each wafer, as mentioned in Section 4.4.1, to remove some higher order wafer-to-wafer variations that were not eliminated in the first standardization.

Fig. 4.5 demonstrates the robust mean and standard deviation of three of the new features: the first dimension in the embedded spaces constructed using Minkowski distance with $p = 1, 2, 3$ as the proximity measure. Each dot in the figure represents the statistics of one wafer. In Fig. 4.5a, most of the wafers have their robust means very close to zero in all three features, and in Fig. 4.5b, the robust standard deviation in the first dimension of the embedded space from Minkowski distance with $p = 1$ and that with $p = 3$ are highly correlated, while

the variation in the dimension of Minkowski $p = 2$ is relatively negligible. More importantly, both Figs. 4.5a and 4.5b show some outliers away from the normal distribution. The wafers with these outlying values have very different characteristics in the new features from the majority of the wafers and should be excluded from statistical analysis.

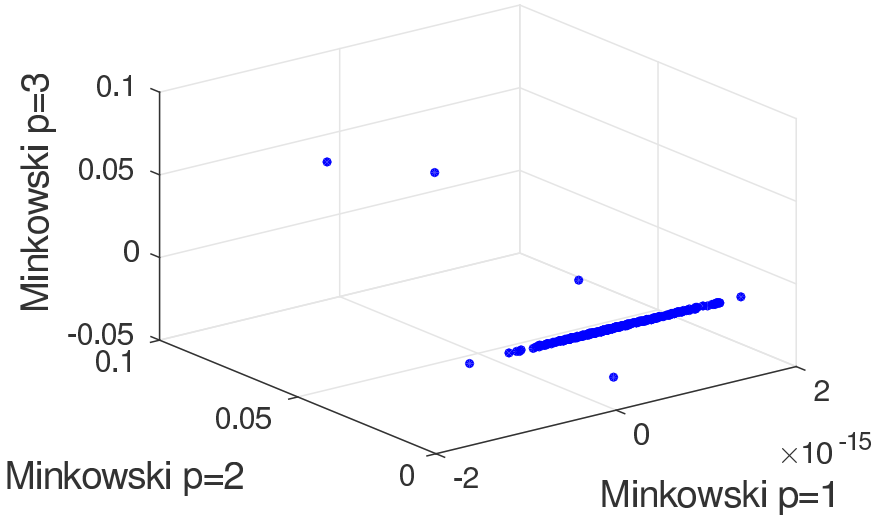
While Fig. 4.5 provides an example to visualize these outliers in three selected features, we can also apply LOF or some simpler outlier analyses such as Mahalanobis distance [41] to quantitatively expose these outlying wafers. Mahalanobis distance is defined as:

$$d_{ab} = \sqrt{(\boldsymbol{x_a} - \boldsymbol{x_b})' \boldsymbol{C}^{-1} (\boldsymbol{x_a} - \boldsymbol{x_b})} \tag{4.15}$$

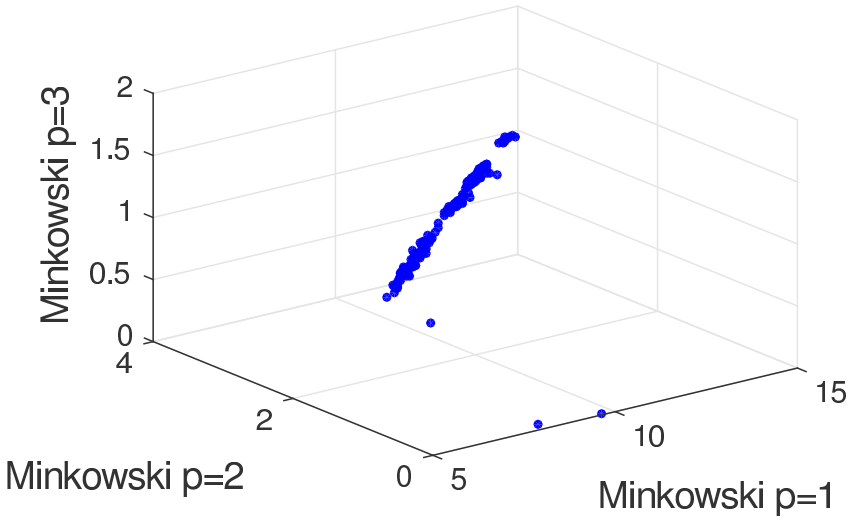
where C is the covariance matrix of the dataset. Intuitively, equation (4.15) computes the distance between two samples in a Euclidean space that is normalized with respect to the covariance matrix of the original Euclidean space, and therefore reveals outliers that has a smaller Euclidean distance to the major population but lies out of the shape of the major population's distribution.

4.4.4 Feature Transformation and Classification

After the generation and standardization of the proximity-based features, we analyze them jointly with the base features for test escape screening. Our objective has been generating *potentially revealing* features without custom investigation for each dataset of which features are really more informative for test escape screening. Our framework creates a general collection of potentially useful features that can



(a) Robust mean



(b) Robust standard deviation

Figure 4.5: The robust mean and standard deviation of each wafer in the feature space of three proximity-based features.

be applied to any dataset/product, which are suitable for known feature reduction and classification algorithms to automatically extract the most useful information out of them for high accuracy classification. In our experiments, we employ *canonical analysis*, proposed in Chapter 2, to the joint feature sets, consisting of the base features and the proximity-based features, for feature reduction. Canonical analysis is a linear transformation which compacts the multi-dimensional separation between classes of samples into the first few dimensions in a transformed canonical space. In our analysis for test escape screening, there are two classes of samples: test escapes (*positive* samples) and good chips (*negative* samples), and compacting the separation in the high-dimensional feature space into a small number of features has been demonstrated to achieve significant runtime reduction and in some cases, greater classification accuracy, based on a conventional classifier such as SVM. Specifically, we apply *C-support vector classification* (C-SVC) provided by LIBSVM [22] as the final classifier. The complete flow of generating the proximity-based features for statistical analysis is illustrated in Fig. 4.6.

4.5 Experimental Results

In this section we present the results of analyzing the proposed proximity-based features jointly with the base features derived in Chapter 3 on a continue-on-fail production test data of an industrial product. We preprocessed the test data to remove confidential information while preserving all information that is relevant to the analysis. The dataset includes more than 700 wafers with 1000+ chips per wafer. We use 200+ wafers as the training set, 200+ wafers as the validation set

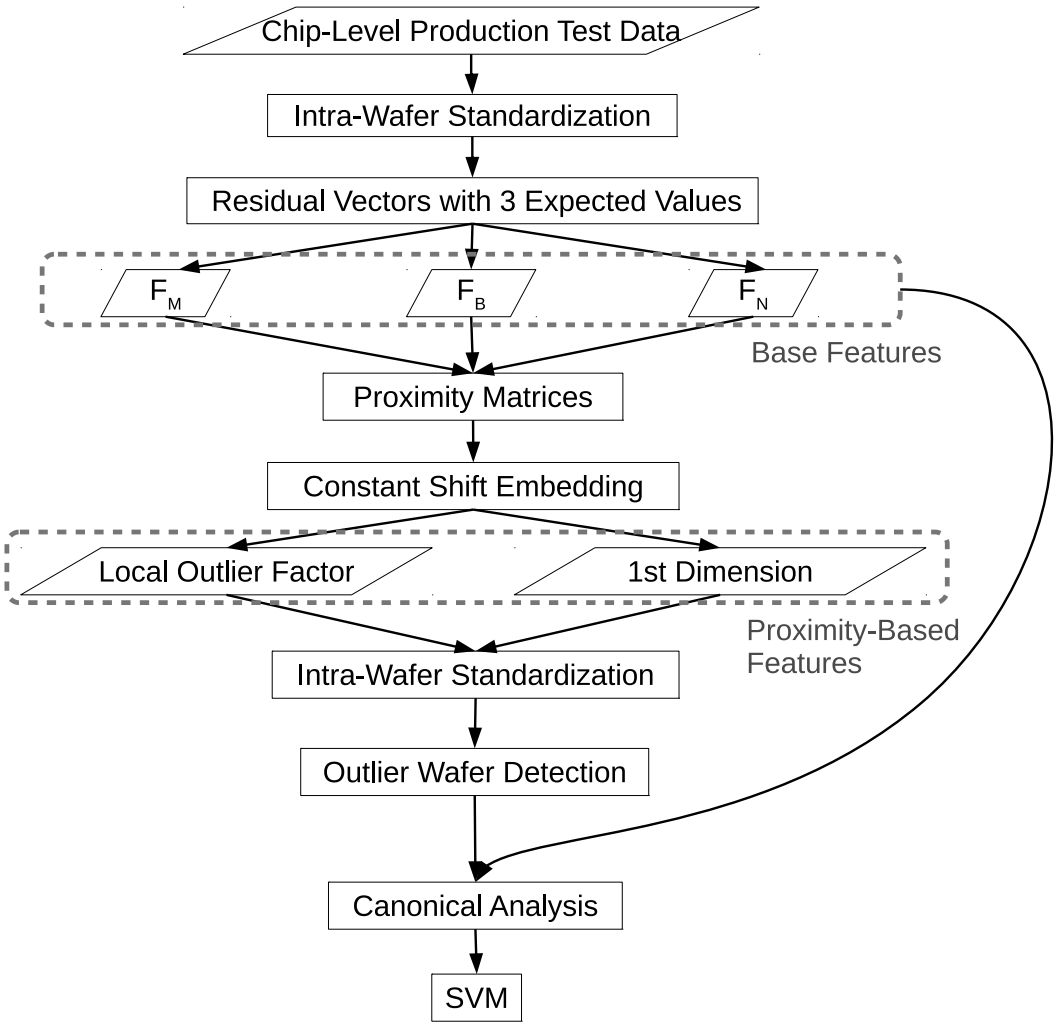


Figure 4.6: The complete flow of generating the proximity-based features for statistical analysis.

for selecting SVM parameters, and the rest 200+ wafers as the testing set. The test program contains more than 200 parametric test items. For our analysis, we emulated the test escape population using the process described in Chapter 3 and derived an emulated test escape population of 560PPM for the testing set.

4.5.1 Classification Accuracy

Fig. 4.7 demonstrates the relative operating characteristics (ROC) curves, i.e. the test escape detection (true positive) rate vs. the yield loss (false positive) rate, of the classification based on the base features with and without the new proximity-based features. The two ROC curves exhibit different trends and cross each other at a yield loss rate of approximately 0.01%. This indicates that including the proximity-based features does provide more information, otherwise the classification accuracy would not be affected. The additional information provided, however, does not generalize from the training set to the testing set and becomes counter-productive at a very low yield loss rate. Given sufficient yield loss rate ($> 0.01\%$), the additional information from the proximity-based features starts to help classify more test escapes and improves the test escape detection rate to 31%, compared with 27% for using the base features alone at a yield loss rate of 0.027%. Therefore, even after the standardizations on the production test data and on the proximity-based features for each wafer, there still exist some significant discrepancies between the training set and the testing set. The cause of such discrepancies requires further investigation and should be removed to improve the consistency between the training set and the testing set.

Fig. 4.8 shows the ROC curves of classification based on the base features plus

different subsets of the proximity-based features. We investigated the results using the features based on the two embedding methods (CSE and kPCA) individually. Similar to Fig. 4.7, the test escape detection rates for using the base features plus the features based on each of the two embedding methods are lower than that using only the base features at a lower yield loss rate. In fact, the classification accuracy based on the base features plus the two kPCA-based features (LOF value and the first dimension of the embedded space) never surpasses the classification accuracy based on the base features only, in the range we searched for an optimal pair of SVM parameters [22]. However, including both subsets of the proximity-based features for classification could lead to a significantly greater test escape detection rate than including either of the subsets alone. In this case, incorporating both subsets of the proximity-based features allows the classification to focus on the additional information that can be effectively generalized to the testing set and be free from the discrepancies between datasets.

4.5.2 Performance Overhead

On average, for one wafer with 1000+ chips and 700+ base features, deriving the pairwise proximity and applying CSE takes 2.4 seconds, while applying LOF takes another 2.2 seconds on an Intel Xeon Quad-core 3.6GHz system. Compared with the runtime for the canonical transform followed by SVM classification, which involves simple linear operations and takes 0.02 second per wafer, the runtime for the nonlinear proximity/distance functions and the LOF algorithm is relatively significant. Moreover, the memory usage and runtime for processing the pairwise proximity grows quadratically with respect to the number of chips per wafer.

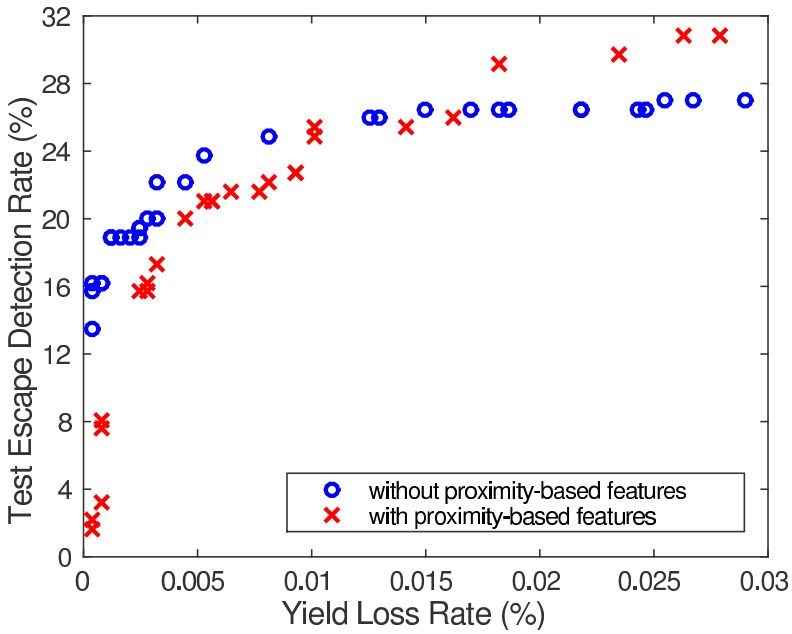


Figure 4.7: The ROC curves of classification based on the base features with and without the proximity-based features.

Therefore, a future direction would be to optimize the algorithms and the flow for generating the proposed features for better efficiency.

In principle, whether it makes sense or not to apply the proximity-based features for statistical tests in addition to the existing base features depends on the cost and quality requirement of the products. For example, including proximity-based features in the analysis may not be cost effective for a high-volume product that requires real-time application of the analysis, e.g. chips for mobile devices. On the other hand, for an extremely quality demanding product that does not require real-time analysis, e.g. processors for centralized servers and chips for safety critical systems, applying the proximity-based features for offline statistical tests could help screen more test escapes without incurring unacceptable extra cost.

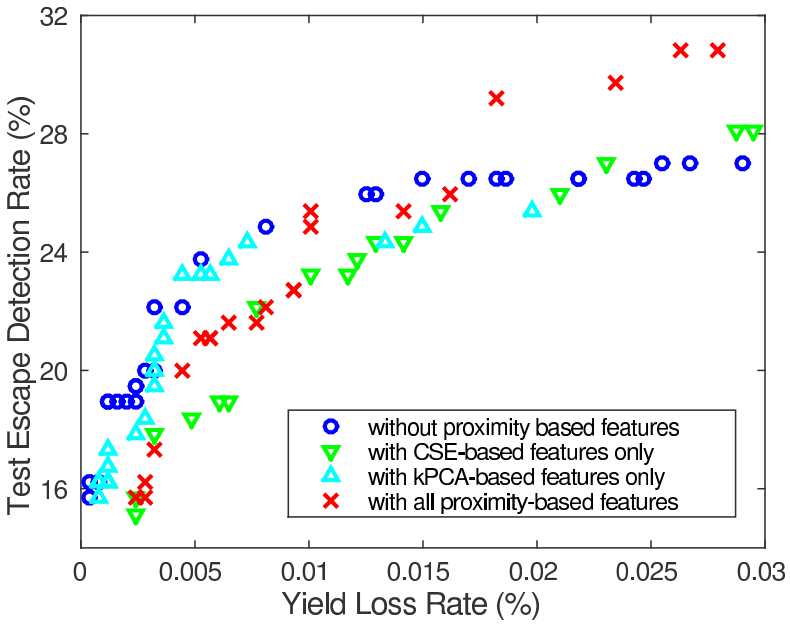


Figure 4.8: The ROC curves of classification based on the base features plus different subsets of the proximity-based features.

4.6 Summary

This chapter proposes a new set of proximity-based features based on a collection of base features: residual vectors with respect to three different expected values of test measurements. We demonstrate a complete flow of generating additional informative features and the reasoning for each step. To expose the abnormalities of test escapes, the proposed method first compares each chip with all other chips on the same wafer in the feature space composed of the base features, followed by constant shift embedding to embed the proximity matrix into an equivalent Euclidean embedding with no distortions. The outlying level of each chip in the embedded space is then converted into a single score using local outlier factor, and the LOF values, jointly with the first dimension of each

embedded space, are used as the new features for test escape screening. The experimental results based on an industrial production test dataset demonstrate that the proximity-based features provide additional information revealing the abnormalities of some test escapes, which further improves the test escape detection rate beyond the state-of-the-art methods that are already comprehensive for test escape detection.

Chapter 5

An Artificial Neural Network Approach

5.1 Introduction

Artificial neural networks (ANNs) have demonstrated great potential and outperformed many other machine learning algorithms in applications such as image and voice recognition. An artificial neural network is composed of an input layer, an output layer, and some *hidden layers*. Each of the layers contains *neurons*, which simulate the biological neurons by summing the weighted values from the input connections and output an activation result based on a selected activation function. Artificial neural networks have potential to learn complex concepts given nonlinear activation functions and multi-layer structures; however, the many choices for designing the structure and the huge number of parameters for training the model are also a challenge for developing an ANN solution.

In this chapter, we propose using a simplified *autoencoder* [42] structure for classifying test escapes. In an autoencoder, the input data layer represents the original features of the sample and the output layer represents the *recovered* features of the sample. The hidden layers usually contain a bottleneck layer, whose number of neurons is smaller than the number of original features. The network from the input layer to the bottleneck layer represents a feature compaction process. Using unsupervised learning, we train the autoencoder with good chips only and set the cost function to be the Euclidean distance between the values in the input and output data layers, so that the autoencoder would derive a smaller number of features that could best represent the features of the good chip population. Based on the trained autoencoder fitting the good chip population, we could then classify a query chip based on its Euclidean distance between the corresponding values in the input and output layers. A test escape is likely to have an abnormally large Euclidean distance.

In our proposed structure, we use only one single hidden layer between the input and output layers and for each neuron, the weighted sum of the input values is directly bypassed to the neuron's output connections without using an activation function. Therefore, the output values are essentially linear combinations of the input values in the proposed structure. We use an industrial production test data to demonstrate that with such a configuration and the chosen cost function, the proposed ANN could achieve higher classification accuracy for test escapes compared with canonical analysis followed by a support vector machine (SVM) classification. It was demonstrated in Chapter 2 that canonical analysis could significantly improve the runtime and, in some cases, the accuracy of a classic SVM

classifier. In Chapter 4, a collection of nonlinear transformations was proposed to generate additional information that further improves the test escape detection rate compared with the framework in Chapter 2. We will demonstrate that the proposed linear ANN also outperforms this framework that incorporates nonlinear information, and significantly reduces the runtime and memory usage required during test application.

In the rest of the chapter, the basic concept of artificial neural networks and the proposed structure will be discussed in Section 5.2 and Section 5.3. Section 5.4 illustrates data processing techniques for generating features that characterize the chips under test. Section 5.5 presents the experimental results, and Section **??** concludes the chapter.

5.2 Artificial Neural Networks

An artificial neural network is composed of multiple neurons. Fig. 5.1 demonstrates an example of an artificial neuron with three input connections and one output connection. The inputs and outputs of the artificial neuron represent the dendrites and axons of an actual neuron. To simulate the excitation reaction of a biological neuron, the weighted sum of the inputs $w_0x_0 + w_1x_1 + w_2x_2$ goes through an activation function $f()$ and the activation result $f(w_0x_0+w_1x_1+w_2x_2)$ is passed to the following neurons. An example of common activation functions is the *sigmoid function*:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.1}$$

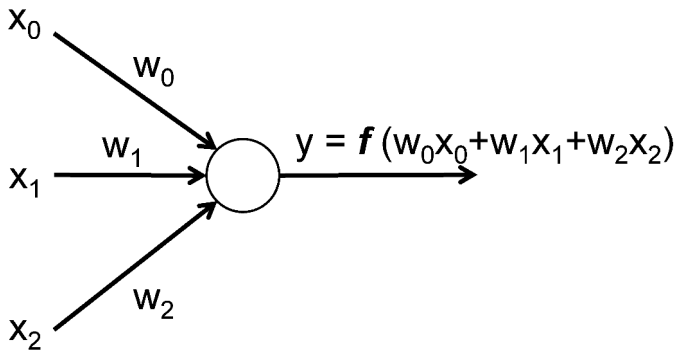


Figure 5.1: An artificial neuron with three inputs and one output. The output of a neuron is the activation result of the weighted sum of the neuron's inputs.

There are three types of layers in an artificial neural network: an input layer, an output layer, and some hidden layers in between, as shown in Fig. 5.2. The structure in Fig. 5.2 is a *feedforward* neural network because no connections between the neurons could form a cycle, otherwise the structure is called a *recurrent* neural network. During the training phase, the input values at each layer are passed to the neurons for calculating the activation results that are passed to the next layer of neurons. The error calculated at the output layer is then used to iteratively update the weights of the neuron connections in each layer backward until the input layer is reached. This process for updating the weights is called *backpropagation*. Through the training phase, a backpropagation algorithm finds a set of weights as the parameters for the model that minimizes the cost function.

5.3 The Proposed Structure

In this study, we use a specific neural network structure, an *autoencoder* [42], for classifying test escapes. In an autoencoder, the input layer and the output layer both represent the original features of the samples. Typically, the number of

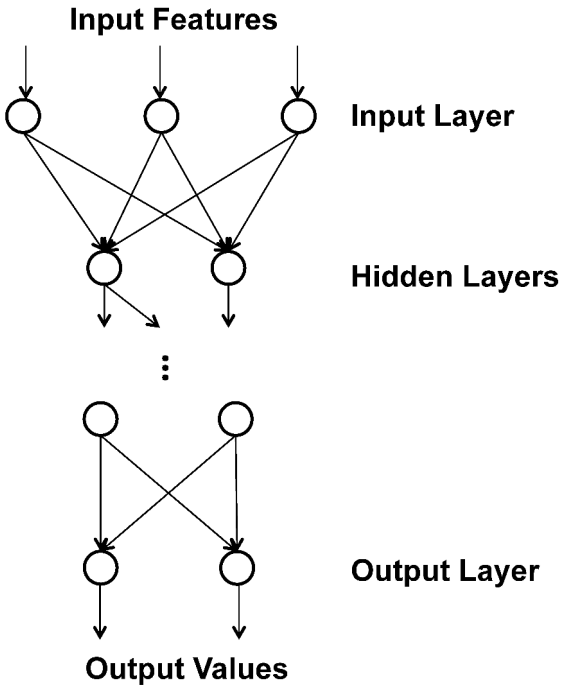


Figure 5.2: A neural network contains an input layer, an output layer, and some hidden layers in between.

neurons in the hidden layers would decrease monotonically from the first hidden layer until reaching a bottleneck layer, in which the number of neurons is smaller than the number of original features. The number of neurons in the hidden layers after the bottleneck layer would then increase monotonically until the last hidden layer is reached, which is a process of recovering the original features. The first half of the autoencoder (for feature compaction) and the second half of the autoencoder (for feature recovering) are usually symmetrical in terms of the number of neurons per layer. During training, a distance between the values in the input and output layers is used as the cost function. Such an autoencoder structure can derive a small number of features that compact the most critical information into the neurons in the bottleneck layer. Recovering the original features from these bottleneck layer features and representing them in the neurons of the output layer helps define a simple cost function for training - the Euclidean distance between the original and recovered features.

In our experiments, we trained multiple autoencoder models with different structures (i.e. the number of neurons in hidden layers and the number of hidden layers). We did not apply activation functions on the neurons because there is no intuitive guideline on what type of information the neural network should focus on. Therefore, the trained autoencoders were essentially linear transformations derived with a unique cost function. The exploration of proper activation functions is part of our future work. The classification accuracy of different autoencoder structures is demonstrated in Section 5.5.1, based on which we selected one structure that achieves the best test escape detection rate at a very low yield loss rate as the classification model. Fig. 5.3 shows the selected structure, whose

structure is given below:

- Each neuron directly passes the weighted sum of its input values to the output without employing an activation function.
- We implement only one hidden layer in the ANN. Let n_{in} and n_{out} be the number of neurons in the input and output layers respectively, and n_h be the number of neurons of the hidden layer, the structure satisfies the following two conditions:

$$n_{in} = n_{out} \tag{5.2}$$

$$n_h < n_{in} \tag{5.3}$$

  For the dataset we analyzed in this chapter, $n_{in} = n_{out} > 700$, and we set $n_h = 500$ empirically.
- The hidden layer and the output layers are both *fully-connected layers*. In a fully-connected layer, a neuron is connected to all neurons in its previous layer.
- The cost function used for training is the Euclidean distance between the corresponding values in the input and output layers.

We use the Caffe package from UC Berkeley [43] for handling and solving for the neural network model, and use *Adam solver* [44] as the backpropagation algorithm. In our experiment, the Adam solver could fit the training data much better and converges faster than the traditional stochastic gradient descent (SGD) method [45]. Details of the Adam solver can be found in [44].

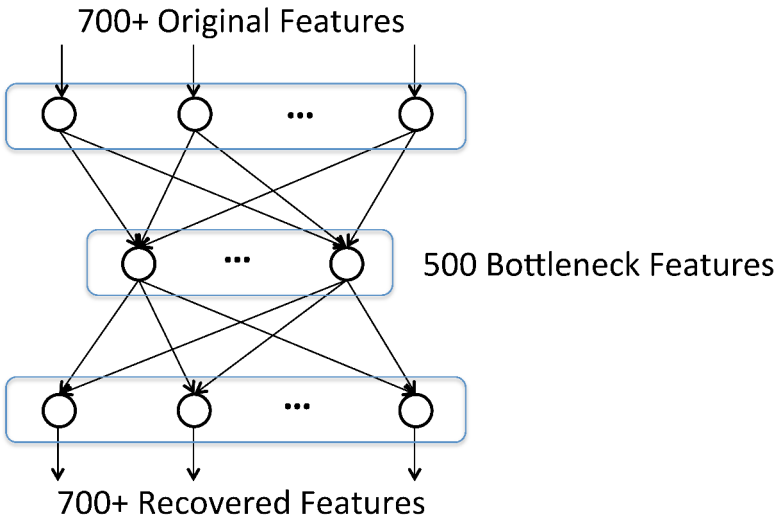


Figure 5.3: The proposed autoencoder structure.

In our analysis, we describe test escape screening as a two-class classification problem - to accurately classify the class of test escapes (positive class) and the class of good chips (negative class). Since test escapes usually have a wide spectrum of root causes and good chips typically have similar performances, it's logical to develop a framework that would expose test escapes as outliers in some aspects so that we can screen them. Therefore we chose to train the autoencoder using good chips only in the training set. In other words, we derive an autoencoder model that only fits the good chips. If a query chip has different characteristics from the good chip population captured in the autoencoder model, the feature values could not be accurately compacted and recovered by the process that was trained using the good chips, and the Euclidean distance between input and output values should be larger than that of a good chip. We therefore use the resulting Euclidean distance of each query chip for determining if it is a test escape.

Without the nonlinear activation function, the feature compaction process in

this structure is effectively a linear transformation. Section 5.5 will demonstrate that this specific structure and cost function could achieve higher classification accuracy than a canonical analysis followed by SVM classification and a collection of nonlinear transformations based on proximity information between each pair of chips on the wafer.

5.4 Feature Processing

In this section we discuss how we standardize the test data and generate features before training the autoencoder.

5.4.1 Data Standardization

Before the test data is used for training or classification, we first standardize the test data for each item on each wafer to the same scale using a method proposed in Chapter 3. This standardization reduces the wafer-to-wafer variation in production test data and therefore is critical for the training and classification accuracy. The test data of each test item for dies on each wafer is standardized to a z-score by:

$$z = \frac{x - \mu}{\sigma} \tag{5.4}$$

where x is the original test measurement, μ is the *robust mean*, and σ is the *robust standard deviation* of the test item. The robust statistics μ and σ are calculated based on the chips on the wafer excluding the outliers, which are found using a general Extreme Studentized Deviate (ESD) test [34].

5.4.2 Feature Generation

In this analysis, we use the *residual vectors* proposed in Chapter 2 as the features to characterize the chips under test. Let M be the number of test measurements in the test program, a chip is characterized by an $M \times 1$ vector $\boldsymbol{r}$:

$$\boldsymbol{r} = \boldsymbol{x_m} - \boldsymbol{x_e} \tag{5.5}$$

where $\boldsymbol{x_m}$ is an $M \times 1$ vector of the measurement values for all test items and $\boldsymbol{x_e}$ is the expected values for the test items.

Defined as the difference between the measurement values and expected values, a residual vector represents how a chip's measurements deviate from those of the normal population. Therefore, residual vectors as the features for analysis capture random variations but remove the effects of systematic variations. Using different expected values, the corresponding residual vectors will reveal unique aspects of the chips under test. We use three expected values proposed in Chapter 3 to generate three distinct types of residual vectors. The three expected values for each test item are: 1) the mean of the measurements for dies on the same wafer, 2) the value predicted based on a bilateral-filtered [18] spatial pattern of the wafer, and 3) the median of the eight closest neighbors' measurements of the query chip.

5.4.3 Proposed Test Flow

Fig. 5.4 shows the proposed flow of using the autoencoder to generate the features and classify test escapes. For each query chip, the Euclidean distance between the values in the input and output layers in the trained autoencoder is

calculated, and a threshold is set on the Euclidean distance for the classification.

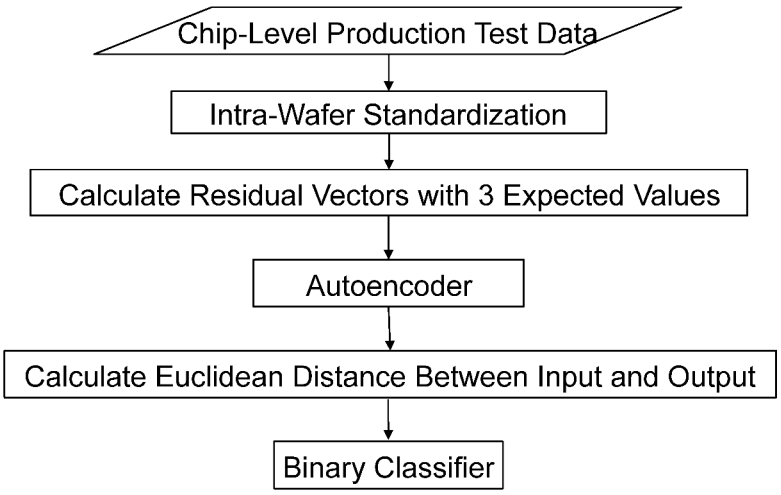


Figure 5.4: The flow of using the proposed autoencoder for test escape screening.

5.5 Experimental Results

In this section we demonstrate the results of analyzing a continue-on-fail industrial production test data. The test data was preprocessed to remove the confidential information while preserving all information that is relevant to the analysis. The dataset includes more than 700 wafers with 1000+ chips per wafer, and there are more than 200 parametric test items in the test program. Based on the 200+ parametric test items, the three residual vectors result in 700+ features for the analysis. We use 200+ wafers as the training set, 200+ wafers as the validation set for selecting parameters of an SVM classifier [22] (for the comparison described in Section 5.5.2), and the rest 200+ wafers as the testing set. For our analysis, we emulated the test escape population using the process described in Chapter 3 and created an emulated test escape population of 560PPM for the

testing set.

5.5.1 Impact of Structure Design

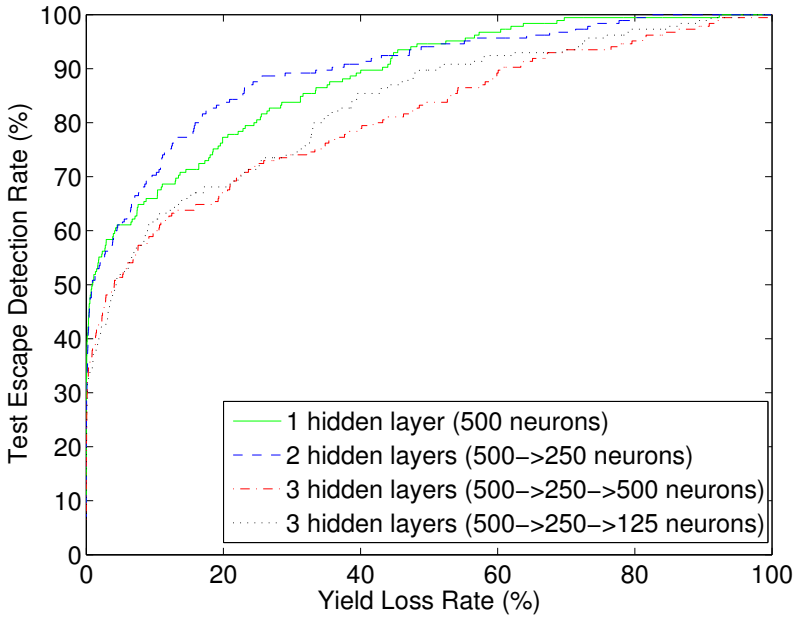
We have tried multiple structure designs for building the autoencoder model. Based on the constraint that the number of neurons in the bottleneck layer should be smaller than that in the input and output layers, we started building the autoencoder with only one hidden layer and empirically set the number of neurons in the hidden layer to be 500, based on the classification accuracy. We then built models with more hidden layers, adding one additional layer at a time while keeping the trained parameters in the existing layers as the initialization for the weights. This iteratively procedure is called *pretraining* [42], which allows the training process to converge to a good solution faster without searching slowly around some local optima.

Fig. 5.5 shows the relative operating characteristics (ROC) curves, which plot the true positive rate (test escape detection rate) versus the false positive rate (yield loss rate), for autoencoder structures with 1) a single hidden layer with 500 neurons, 2) a hidden layer with 500 neurons followed by another hidden layer with 250 neurons, 3) three hidden layers with the numbers of neurons in each being 500, 250, 500, respectively, and 4) three hidden layers with the numbers of neurons in each being 500, 250, 125, respectively.

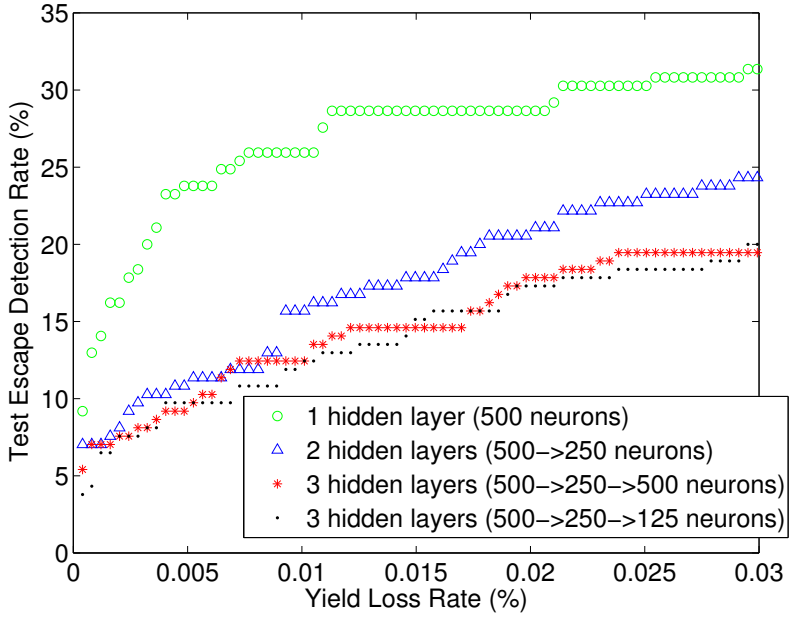
In Fig. 5.5a, the two-layer structure could detect more test escapes than the single-layer structure at a yield loss rate between 5% and 45%. Having three layers in the structure, however, decreases the test escape detection rate at any given yield loss rate compared with the structure with only one or two layers. Although

we did not apply activation functions for the neurons, which means each layer of the autoencoder is essentially a linear transformation and therefore each autoencoder with multiple hidden layers has an equivalent structure with only one hidden layer (imagine multiplying all the transform matrices representing each hidden layers to obtain a single transform matrix), the structures with different numbers of layers still result in very different ROC curves because the backpropagation algorithm for updating the weights is impacted by the structure of the autoencoder.

Fig. 5.5b plots the same ROC curves in the region with very low yield loss rate, which is usually required for the application of test escape screening. Given the very low yield loss rate, the structure with a single hidden layer significantly outperforms the other structures. Although a structure with more hidden layers and neurons have the potential to learn more complicated characteristics, the learned characteristics of the good chip population based on the unsupervised learning does not necessarily help detecting the test escapes. In other words, the additional learned characteristics of the good chips, if any, may not be unique to the good chips and therefore does not help expose test escapes as anomalies because we did not specify any characteristics of the test escapes during the training phase. In this dataset, the simplest structure with only one hidden layer have modeled the most critical characteristics of the good chips that could be used to identify test escapes in the target region of the yield loss rate.



(a) The ROC curves of different structure designs of the autoencoder.



(b) The ROC curves in the target yield loss rate region.

Figure 5.5: The ROC curves demonstrate the classification accuracy for different structure designs of the autoencoder.

5.5.2 Classification Accuracy

Fig. 5.6 shows the ROC curves of three frameworks for comparison. In the first framework (proposed in Chapter 2), shown in green triangles, the three types of residual vectors were used jointly as the input features for a canonical analysis followed by a support vector machine (SVM). Canonical analysis is a linear transformation that compact the separation between classes in the high-dimensional feature space into the first few dimensions in the transformed feature space. It has been demonstrated in Chapter 2 that applying canonical analysis before SVM for feature reduction can significantly improve the runtime and in some cases the accuracy for classifying test escapes. In the second framework (proposed in Chapter 4), shown in blue circles, pairwise proximity features were calculated in the feature space composed of the three types of residual vectors, and then used jointly with the three types of residual vectors for canonical analysis followed by SVM. The proximity features were generated by applying multiple nonlinear distance/proximity functions on each pair of chips on the same wafer, and the generated nonlinear information could reveal additional test escapes compared with existing linear transformation methods at a cost of excessive computation time and memory usage. The classification accuracy of the proposed autoencoder framework is marked by the red dots.

As discussed in Chapter 4, including the proximity features for the method combining canonical analysis and SVM could improve the test escape detection rate for a yield loss rate being greater than 0.016%, but could degrade the test escape detection rate at a lower yield loss rate, compared with the same method without using these non-linear features. The difference between their ROC curves

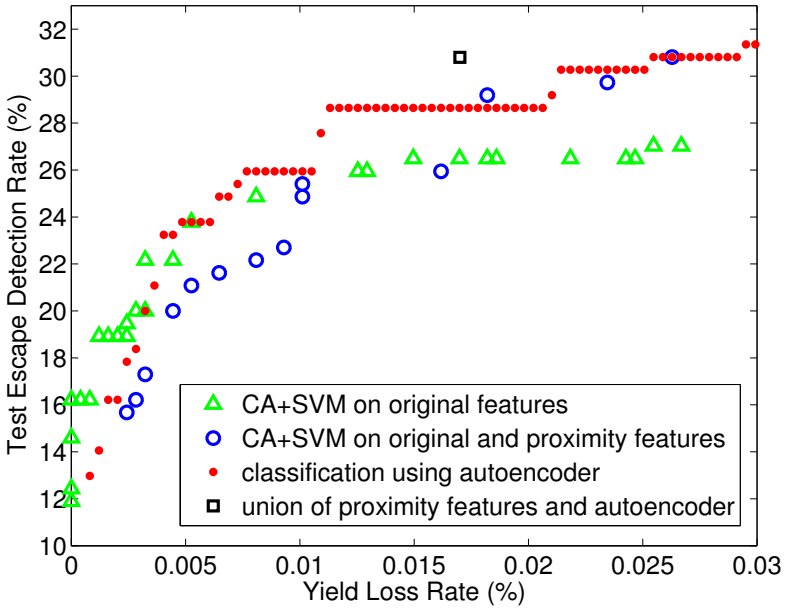
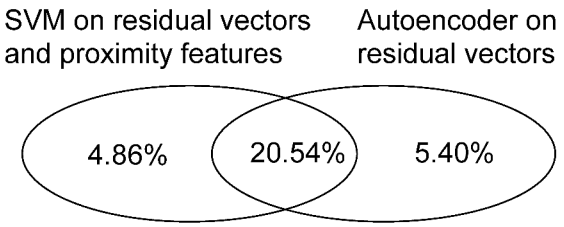


Figure 5.6: The ROC curves of three frameworks.

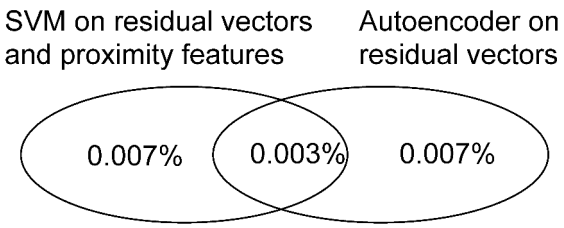
indicates that including the proximity features does provide more relevant information for classification in the training set; otherwise the trained model would not be different. However, the additional information for detecting test escapes in the training set could not be generalized to the testing set when the yield loss rate is low, say, below 0.016%. On the other hand, the classification using autoencoder could detect more test escapes than the first framework when the yield loss rate is greater than 0.005%. Compared with the second framework in which the proximity features are included for analysis, classification using autoencoder consistently detects more test escapes when the yield loss rate is below 0.018% and the detection rates of the two classifications become similar at a greater yield loss rate.

Analysis for the sets of the detected test escapes at a yield loss rate of 0.01% by

the second framework and the autoencoder is summarized in Fig. 5.7. The autoencoder could reveal similar amount of test escapes compared with the framework that incorporates a collection of nonlinear transformations at this yield loss rate. Each of these linear and nonlinear frameworks, however, detects a unique subset of the test escapes - the autoencoder could uniquely detect 5.40% of the test escape population while the framework utilizing the nonlinear transformations could uniquely detect 4.86% of the test escapes. Out of the 0.01% yield loss population, 0.003% was caused by both methods. Taking the union of the two methods' results, we could achieve a test escape detection rate of 30.8% at a yield loss rate of 0.017%, which is better than either of the two methods alone, as marked by the black square in Fig. 5.6.



(a) The test ecsape detection rate by the two methods.



(b) The yield loss rate by the two methods.

Figure 5.7: The Venn diagrams of the test escape and yield loss populations resulted from the SVM on proximity features and residual vectors (the method in Chapter 4) and from the proposed autoencoder.

5.5.3 Trained Parameters in the Model

We conducted further analysis of the autoencoder structure to gain useful insights for better understanding of the classification process and for diagnosis of the test escapes.

Fig. 5.8 shows the distribution of the values of the trained weights in the hidden layer (i.e. weights on the connections from the input layer to the hidden layer). Recall that there are more than 700 original features in the input layer and 500 neurons in the hidden layer, and that the hidden layer is a fully-connected layer, so there are more than 350000 weights in this layer and more than 700000 weights to be optimized during training in the entire structure. In Fig. 5.8, most of the weights are smaller than 0.5 and centered at 0.

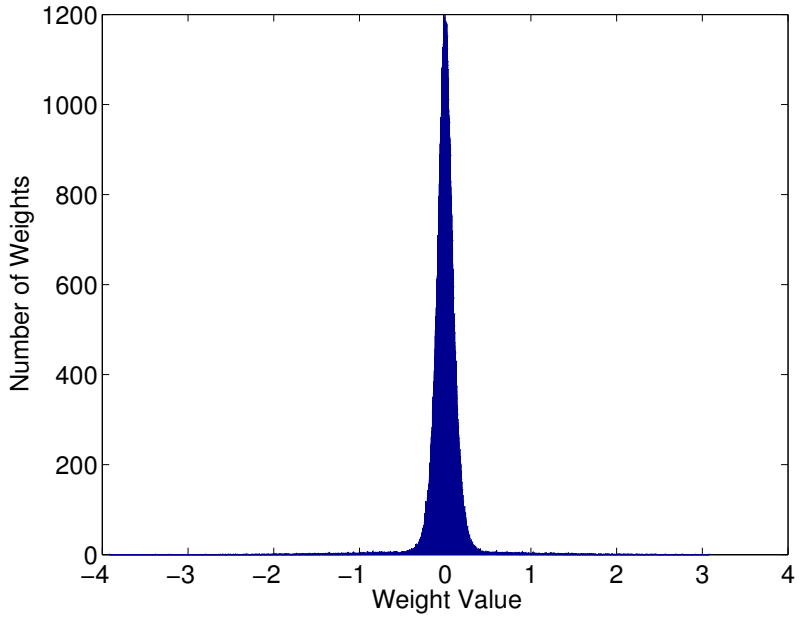


Figure 5.8: The histogram of the trained weights in the hidden layer.

The absolute values of the trained weights in the hidden layer are demonstrated

in Fig. 5.9 as a color-coded map, in which the horizontal axis corresponds to the neuron index in the source of the connections (the input layer) and the vertical axis corresponds to the neuron index in the destination of the connections (the hidden layer). There are clearly some vertical and horizontal stripe patterns in the weights, e.g. a bright vertical stripe around source neuron index 200. This means that these neurons in the first layer, which represents the original features, are more critical in the derived linear system. A dark horizontal stripe around neuron index 175, for example, means that these neurons in the hidden layers are relatively less important than the others. Fig. 5.10, showing the sums of the absolute values of the weights for each column in Fig. 5.9, demonstrates the relative importance of the input neurons (original features). Fig. 5.11 sorts the values in Fig. 5.10, and it shows that most of the original features have similar significance with a sum of absolute weights between 40 to 60. Those original features with a value larger than 60 in Fig. 5.11 are more important in the trained linear system. Such information could indicate which features (test items) are more important for feature compaction and recovering, and if a test escape is screened, what features of the test escape are more likely to be different from the good chip population.

Another aspect of the trained model is shown in Fig. 5.12, in which the pairwise correlations between columns of weights in Fig. 5.9 are plotted in a logarithmic scale. Fig. 5.12 is symmetric, i.e. values at location (i, j) and location (j, i) are identical. A larger value in Fig. 5.12 means that the corresponding two neurons in the input layer, which represent the original features, have similar patterns in the trained weights on the connections to the hidden layer. The cross pattern in

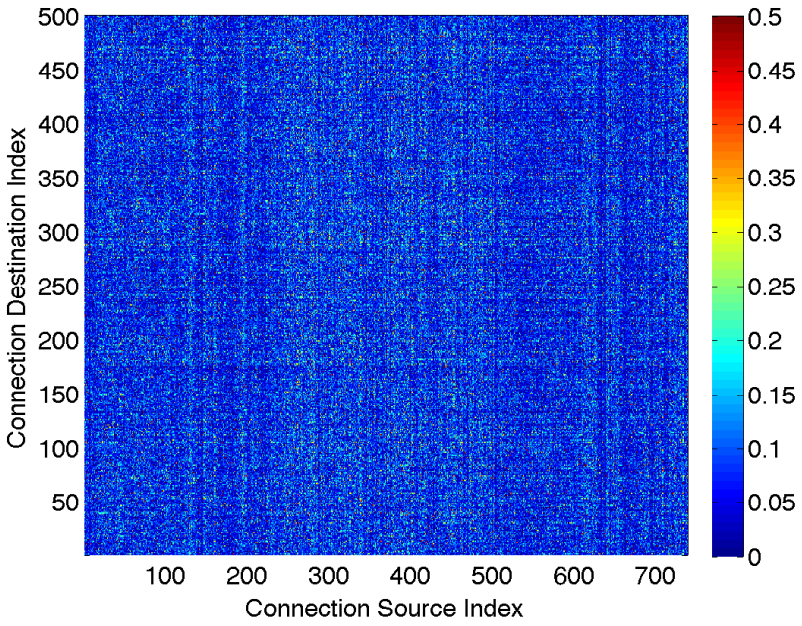


Figure 5.9: The absolute values of the weights in the hidden layer as a color-coded map.

the middle of Fig. 5.12 could also be analyzed with domain-specific knowledge to better understand the underlying relations among the test items.

5.5.4 Performance Comparison

On an Intel Xeon Quad-core 3.6GHz system, the classification in the first framework using canonical analysis and SVM takes 0.02 seconds per wafer, the second framework takes 4.6 seconds per wafer for generating the proximity features and classification, and the autoencoder takes 0.1 seconds for the classification per wafer. Compared with the second framework in which a collection of nonlinear transformation are applied, which incurs a significant amount of runtime and memory usage, the linear classification using the autoencoder could reduce the

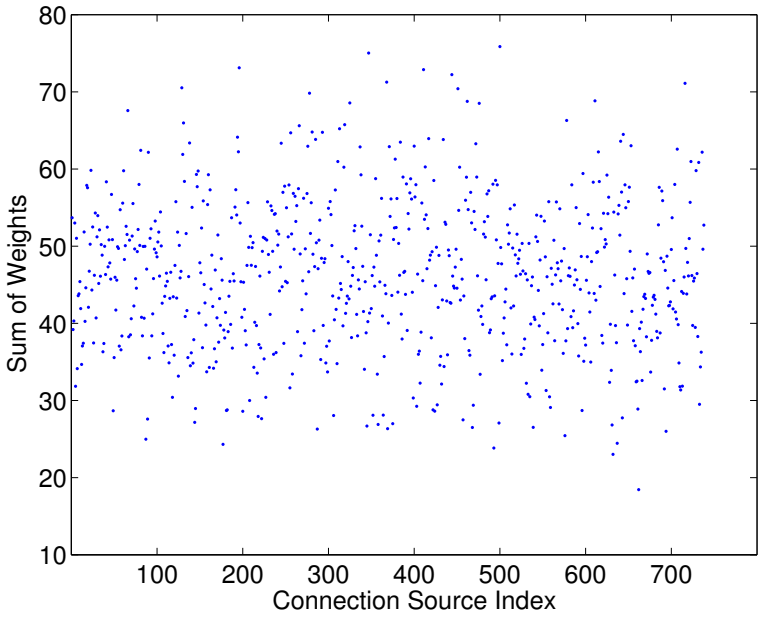


Figure 5.10: The vertical sum of the absolute values of the weights in the hidden layer.

runtime by 46X and achieve a higher classification accuracy.

5.6 Summary

In this chapter, we propose an autoencoder structure that could classify test escapes more accurately than the state-of-the-art statistical and machine learning approaches proposed in Chapter 2 and 4. The specific structure and the cost function of the autoencoder, though only a linear transformation, could reveal even more test escapes compared with a framework incorporating a collection of nonlinear transformations in Chapter 4.

One constraint in this structure is that the number of neurons in the hidden layer must be smaller than the number of original features in the input/output

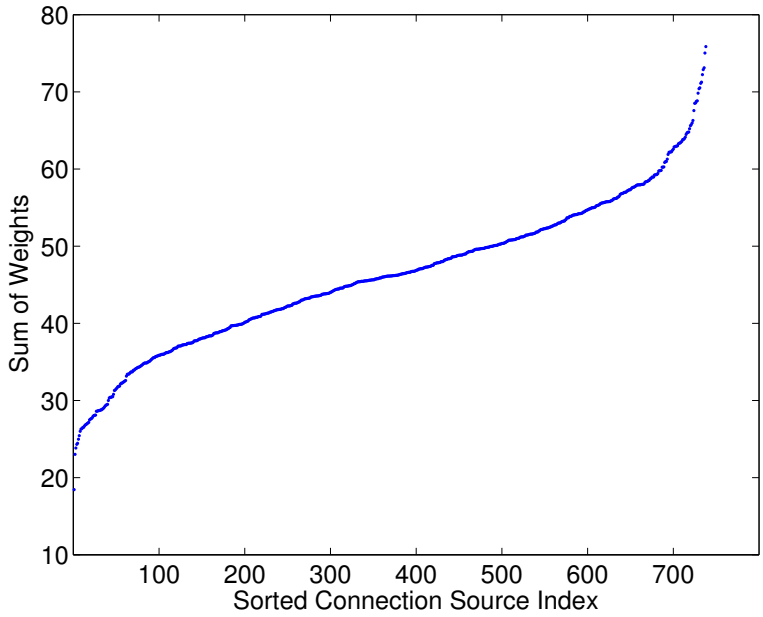


Figure 5.11: The sorted sum of the absolute values of the weights in the hidden layer.

layer. If the number of neurons in the hidden layer is greater than the number of original features, the autoencoder could fit the training data better (i.e. resulting in smaller Euclidean distance between the input and output layers for the training set). However, such a structure would converge to a model that directly bypasses the values from the input to the output, therefore loses its ability to distinguish test escapes from the good chips because the model trained this way could fit any query chip, including test escapes, well.

We tried multiple structure designs for the autoencoder and selected one that could identify the most test escapes in the target region of the yield loss rate. The autoencoder can be viewed as a noise removal process, which keeps only the essential, unique characteristics of the good chip population through the feature

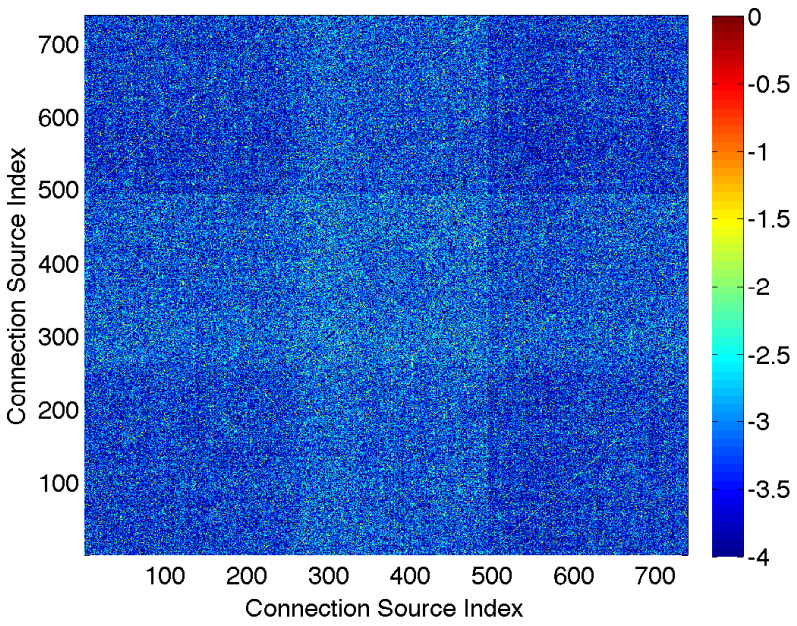


Figure 5.12: The correlation between each two columns of absolute values of the weights. Note that the values are in logarithmic scale.

compaction and recovery process. However, since the training process is unsupervised, the model does not necessarily learn characteristics that could distinguish test escapes from the good chips. Therefore, after building the models it is important to select the one with highest classification accuracy based on some validation dataset.

In addition to the current configuration of the autoencoder, there are still many possible structures, e.g. the choice of the activation functions, the cost function, and the solver for updating the weights. An optimal configuration of the structure for maximizing the test escape detection rate remains part of our future work.

Chapter 6

Conclusion

This research explores machine learning techniques for test escape screening based on semiconductor production test data. Since in machine learning applications, having revealing features often has greater impact on the performance than the selection of classification algorithms, we focus more on feature engineering for extracting more information from the given test data. Our general guideline for the research is to include as many potentially useful features as possible and apply machine learning algorithms that automatically extract the most useful information for classification. The set of potentially useful feature sets could therefore be applied to multiple products or datasets without domain-specific knowledge and the machine learning algorithm would identify the most critical information for the specific dataset that is being analyzed.

For creating the collection of potentially useful feature sets, we propose using the residual vectors with three different expected values: the mean of the measurements on the wafer, the bilateral filtered spatial pattern of the wafer, and

the median of the eight closest neighbors' measurements, which results in three unique feature sets. Pairwise proximity is also proposed as nonlinear transformations based on the three potentially useful feature sets as a post processing for generating additional features that reveal more abnormalities of the test escapes.

A linear transformation, canonical analysis, is proposed for effective feature reduction for the collection of potentially useful features. AdaTest, in which only the most critical features need to be produced during test application, is proposed to significantly reduce the runtime and memory usage compared with the framework of canonical analysis followed by SVM. In addition, an autoencoder classification is developed, which demonstrates the potential of artificial neural networks for test escape screening.

While we have introduced and developed multiple machine learning frameworks for screening test escapes and demonstrated their effectiveness, there are still issues in the machine learning frameworks that could be improved. For example, there still exist discrepancy between the training set and the testing set even after carefully removing the wafer-to-wafer variation and trying to use only the characteristics that are free from temporal variations as the features for analysis. There could be potentially more powerful machine learning solutions such as different ANN structures other than the autoencoder structure we propose. The research could really benefit from more shared industrial data with real test escape information, for exploring additional features and evaluating different machine learning techniques.

Bibliography

[1] P. M. O'Neill, *Statistical test: A new paradigm to improve test effectiveness & efficiency*, in *Proc. Int'l Test Conf. (ITC)*, Oct., 2007.

[2] N. Sumikawa, J. Tikkanen, L.-C. Wang, L. Winemberg, and M. S. Abadir, *Screening customer returns with multivariate test analysis*, in *Proc. Int'l Test Conf. (ITC)*, Nov., 2012.

[3] H. H. Chen, R. Hsu, P. Yang, and J. J. Shyr, *Predicting system level test and in field customer failures using data mining*, in *Proc. Int'l Test Conf. (ITC)*, Sept., 2013.

[4] Automotive Electronics Council, *Guidelines for part average testing*, .

[5] P. M. O'Neill, *Production multivariate outlier detection using principal components*, in *Proc. Int'l Test Conf. (ITC)*, Oct., 2008.

[6] K. M. Butler, S. Subramaniam, A. Nahar, J. M. C. Jr., and T. J. Anderson, *Successful development and implementation of statistical outlier techniques on 90nm and 65nm process driver devices*, in *Proc. Int'l Reliability Physics Symp.*, Mar., 2006.

[7] B. E. Stine, D. S. Boning, and J. E. Chung, *Analysis and decomposition of spatial variation in integrated circuit processes and devices*, *IEEE Trans. on Semiconductor Manufacturing* **10** (1997), no. 1 24–41.

[8] X. Li, R. Rutenbar, and R. Blanton, *Virtual probe: A statistically optimal framework for minimum-cost silicon characterization of nanoscale integrated circuits*, in *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD)*, Oct., 2009.

[9] N. Kupp, K. Huang, J. Carulli, and Y. Makris, *Spatial estimation of wafer measurement parameters using gaussian process models*, in *Proc. Int'l Test Conf. (ITC)*, Nov., 2012.

[10] A. Nahar, K. Butler, J. Carulli, and C. Weinberger, *Quality improvement and cost reduction using statistical outlier methods*, in *Proc. IEEE Int'l Conf. on Computer Design (ICCD)*, Sept., 2009.

[11] W. C. Riordan, R. Miller, and E. R. S. Pierre, *Reliability improvement and burn in optimization through the use of die level predictive modeling*, in *Proc. IEEE Int'l Reliability Phys. Symp.*, Apr., 2005.

[12] N. Sumikawa, L.-C. Wang, and M. S. Abadir, *A pattern mining framework for inter-wafer abnormality analysis*, in *Proc. Int'l Test Conf. (ITC)*, Sept., 2013.

[13] R. A. Johnson and D. W. Wichern, *Applied Multivariate Statistical Analysis*. Prentice Hall, fifth ed., 2002.

[14] S. M. Scheiner, *Multiple response variables and multispecies interactions*, in *Design and Analysis of Ecological Experiments*, ch. 6. Oxford Univ. Press, second ed., 2001.

[15] H.-M. Chang, K.-T. Cheng, W. Zhang, X. Li, and K. Butler, *Test cost reduction through performance prediction using virtual probe*, in *Proc. Int'l Test Conf. (ITC)*, Sept., 2011.

[16] C.-K. Hsu, F. Lin, K.-T. Cheng, W. Zhang, X. Li, J. M. Carulli Jr., and K. M. Butler, *Test data analytics - exploring spatial and test-item correlations in production test data*, in *Proc. Int'l Test Conf. (ITC)*, Sept., 2013.

[17] S. Zhang, F. Lin, C.-K. Hsu, K.-T. Cheng, and H. Wang, *Joint virtual probe: Joint exploration of multiple test items' spatial patterns for efficient silicon characterization and test prediction*, in *Proc. Conf. Design, Automation, and Test in Europe (DATE)*, Mar., 2014.

[18] C. Tomasi and R. Manduchi, *Bilateral filtering for gray and color images*, in *Proc. IEEE Int'l Conf. Computer Vision*, Jan., 1998.

[19] H. Hotelling, *Relations between two sets of variates*, *Biometrika* **28** (1936) 321–377.

[20] L. Ning, A. Nahar, W. R. Daasch, K. M. Butler, J. M. C. Jr., and S. Subramaniam, *Burn-in reduction using robust canonical correlation analysis*, in *Proc. SRC TECHCON*, Oct., 2005.

[21] R. A. Fisher, *The use of multiple measurements in taxonomic problems*, *Annals of Eugenics* **7** (1936) 179–188.

[22] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, *ACM Trans. on Intelligent System and Technology* **2** (2011) 27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[23] N. Sumikawa, D. Drmanac, L.-C. Wang, L. Winemberg, and M. Abadir, *Forward prediction based on wafer sort data - a case study*, in *Proc. Int'l Test Conf. (ITC)*, Sept., 2011.

[24] S. Krishnan and H. G. Kerkhoff, *Exploiting multiple mahalanobis distance metrics to screen outliers from analog product manufacturing test responses*, *IEEE Design & Test* **30** (2013), no. 3 18–24.

[25] W. R. Daasch, J. McNames, D. Bockelman, and K. Cota, *Variance reduction using wafer patterns in IddQ data*, in *Proc. Int'l Test Conf. (ITC)*, Oct., 2000.

[26] W. R. Daasch, K. Cota, and J. McNames, *Neighbor selection for variance reduction in IDDQ and other parametric data*, in *Proc. Int'l Test Conf. (ITC)*, Oct., 2001.

[27] R. Madge, B. H. Goh, V. Rajagopalan, C. Macchietto, W. R. Daasch, C. Schuermyer, C. Taylor, and D. Turner, *Screening minVDD outliers using feed-forward voltage testing*, in *Proc. Int'l Test Conf. (ITC)*, Oct., 2002.

[28] R. Madge, M. Rehani, K. Cota, and W. R. Daasch, *Statistical post-processing at wafersort-an alternative to burn-in and a manufacturable solution to test limit setting for sub-micron technologies*, in *Proc. IEEE VLSI Test Symp. (VTS)*, May, 2002.

[29] W. R. Daasch and R. Madge, *Variance reduction and outliers: Statistical analysis of semiconductor test data*, in *Proc. Int'l Test Conf. (ITC)*, Nov., 2005.

[30] P. Viola and M. J. Jones, *Robust real-time face detection*, *Int'l Jour. Computer Vision* **57** (May, 2004) 137–154.

[31] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, *Jour. Computer and System Sciences* **55** (Aug., 1997) 119–139.

[32] C. Bishop, *Pattern Recognition and Machine Learning*. Prentice Hall, 2007.

[33] B. Rosner, *Percentage points for a generalized esd many-outlier procedure*, *Technometrics* **25** (May, 1983) 165–172.

[34] NIST/SEMATECH, *e-handbook of statistical methods*, *http://www.itl.nist.gov/div898/handbook/* (2003).

[35] S. Sabade and D. M. H. Walker, *Improved wafer-level spatial analysis for IDDQ limit setting*, in *Proc. Int'l Test Conf. (ITC)*, Oct., 2001.

[36] V. Roth, J. Laub, M. Kawanabe, and J. Buhmann, *Optimal cluster preserving embedding of nonmetric proximity data*, *IEEE Trans. on Pattern Analysis and Machine Intelligence* **25** (Dec, 2003) 1540–1551.

[37] B. Schölkopf, A. J. Smola, and K.-R. Müller, *Advances in Kernel Methods*, ch. Kernel Principal Component Analysis, pp. 327–352. MIT Press, 1999.

[38] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, *LOF: Identifying density-based local outliers*, in *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pp. 93–104, 2000.

[39] C. Cortes and V. Vapnik, *Support-vector networks*, *Machine Learning* **20** (Sept., 1995) 273–297.

[40] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[41] P. C. Mahalanobis, *On the generalised distance in statistics*, in *Proc. Nat. Inst. Sci. India*, pp. 49–55, 1936.

[42] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, *Science* **313** (2006), no. 5786 504–507.

[43] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, *Caffe: Convolutional architecture for fast feature embedding*, *arXiv preprint arXiv:1408.5093* (2014).

[44] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in *International Conference for Learning Representations*, July, 2015.

[45] L. Bottou, *Stochastic gradient descent tricks*, *Neural Networks: Tricks of the Trade* (2012).