AdaTest: An Efficient Statistical Test Framework for Test Escape Screening

Fan Lin, Chun-Kai Hsu, and Kwang-Ting Cheng
Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106

***Abstract*—Statistical analyses based on production test data can help identify test escapes, which are chips that pass the test program but fail later at system-level test or in field. Such analyses do not require extra physical measurements and can be referred to as statistical tests. For designing effective statistical tests, this paper investigates the use of a learning framework based on Adaptive Boosting, which has demonstrated great success in real-time face and object recognition. The framework is composed of a cascade of AdaBoost classifiers, each of which uses a small set of most relevant features that are automatically selected in the training phase, to identify a subset of test escapes. This framework therefore generates only the features that are most relevant for classification and significantly reduces the runtime and memory usage for statistical tests during test application. We also propose a new feature set to characterize the chips under test and demonstrate that including the new feature set as input to the proposed feature selection framework could reveal more test escapes.**

I. Introduction

The size and complexity of production test programs have been growing to include more tests and design/process corners to address the demanding quality requirements of modern chips. In addition to the pass/fail results of each test item, there exist multiple types of correlations in test data [1]. Statistical tests [2], 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 [3] and Sumikawa *et al.* [4] investigated the distribution of the population under test using principal component analysis (PCA) on correlated test items, and Krishnan and Kerkhoff [5] explored multiple Mahalanobis distances as the metric for screening. Chen *et al.* [6] also demonstrated various data mining techniques in predicting system-level test (SLT) failures.

Daasch *et al.* [7] 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 [8]–[11]. Lately, Lin *et al.* [12] 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. In their study, 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. The authors showed that 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 paper proposes a new statistical test framework that adopts some of the key ideas behind the popular Viola and Jones [13] framework, which was originally designed for real-time face recognition. Our proposed framework, named *AdaTest*, contains a cascade of AdaBoost [14] 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, in our experiment we demonstrate that AdaTest and an existing method [12], 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.

Another contribution of this paper is that we introduce a new residual vector in addition to the two residual vectors used in [12]. 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

978-1-4673-6578-9/15/$31.00 ©2015 IEEE

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 paper is organized as the following. Section II illustrates the AdaTest framework, Section III discusses the data preparation and feature generation for the framework, and Section IV demonstrates experimental results. Section V concludes the paper.

II. 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.

A. AdaBoost

Adaptive boosting, or *AdaBoost* [14], 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 [13], 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 [15] 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.

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

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

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.

B. 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. 1.

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

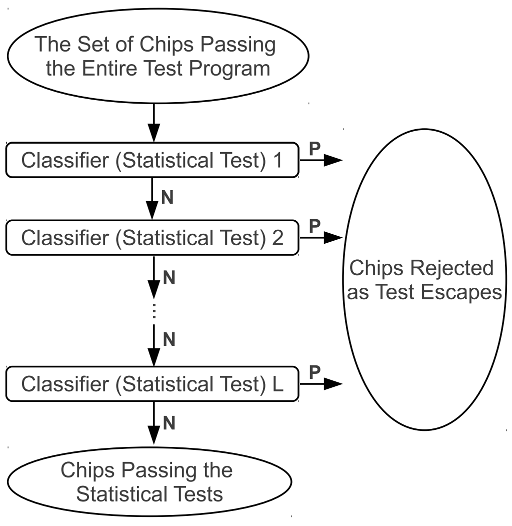

The Set of Chips Passing the Entire Test Program
Classifier (Statistical Test) 1
P
N
Classifier (Statistical Test) 2
P
N
N
Classifier (Statistical Test) L
P
N
Chips Rejected as Test Escapes
Chips Passing the Statistical Tests

Fig. 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.

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.

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
```

III. Data Preparation and Feature Generation

A. 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* 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:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

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 [16]. 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 paper and can be found in [17]. 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.

B. Features for Classification

In this study we characterize the chips using the *residual vectors* proposed in [12]. Each chip with M test measurements is characterized by an $M \times 1$ residual vector $\mathbf{r}$:

$$\mathbf{r} = \mathbf{x_m} - \mathbf{x_e} \tag{2}$$

where $\mathbf{x_m}$ is an $M \times 1$ vector of the measured values and $\mathbf{x_e}$ 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 [12], 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 [19], [20] and Gaussian process model [21] 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 [7], [22], [23]. 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. 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 [7]. Section IV 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 paper, 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 [8].

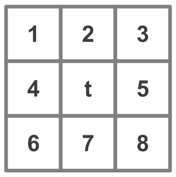


Fig. 2: The median among the measurements of eight neighbors is used as the expected value for the target chip t in the middle.

IV. 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.

A. Emulating Test Escapes

Similar to the setup proposed in [12], the idea is to emulate test escapes using intrinsically defective chips which have subtle syndromes in their test measurements. We first identified faulty chips in the dataset that failed only one test item (and passed all the other test items in the continue-on-fail test program) and considered them as intrinsically defective, but not catastrophic failures. We then *hid* the failing measurement for each of these failing chips by replacing it with the median value of its corresponding test item among the population of good chips in the same wafer, while keeping the measurement values of all other non-failing test items intact. After such manipulation, the modified measurement data of these faulty chips would pass the entire test program, while their measurement values (except for the modified item) still contain the syndromes caused by their intrinsic defects.

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.

B. Classification Accuracy

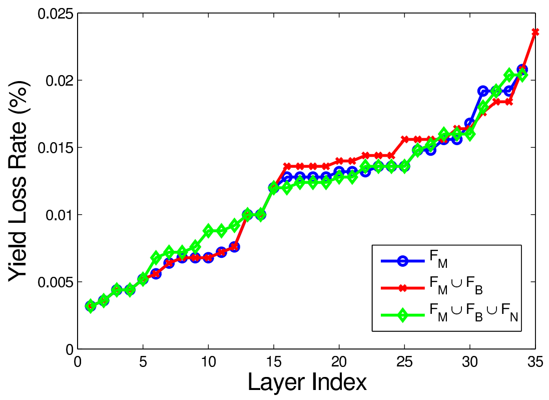
For simplicity and clarity, we use the following notations for the three feature sets we derived in Section III-B:

- 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

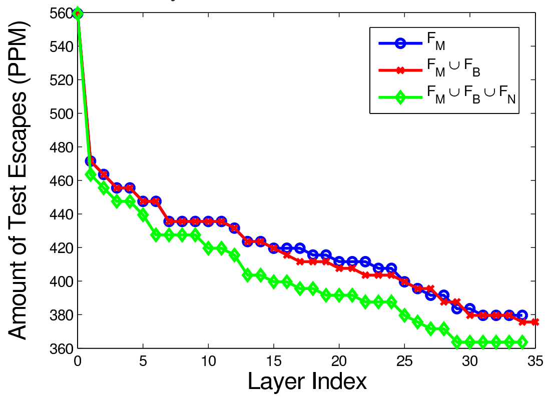
Fig. 3 demonstrates the classification accuracy of the cascaded classifiers on the testing set, with y_{layer} set to 0.01%, based on three choices of the input features for classification. Let M be the number of test items, the three choices are:

- 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. 3a shows that the accumulated yield loss rate increases as more layers of AdaBoost classifiers are added to the cascade, and Fig. 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 [12], using $F_M \cup F_B$ as the input features resulted in significantly higher classification accuracy



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



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

Fig. 3: The accumulated yield loss rate and the amount of remaining undetected test escapes versus the number of layers in the cascade of classifiers.

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. 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 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.

Fig. 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

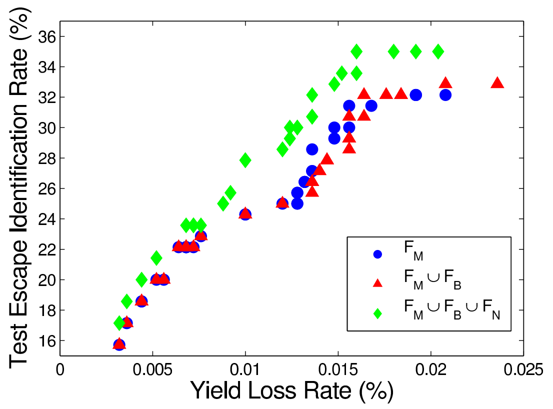


Fig. 4: The ROC curves of classification based on different choices of input features.

certain yield loss rates.

Lin *et al.* [12] proposed an SVM-based framework which incorporates multiple features and transforms them into a canonical space for feature reduction, in which the separation between the classes of populations is maximized in the first few dimensions after the features are transformed into a canonical space. The experimental results in [12] demonstrated significant classification accuracy improvement compared with PCA. Fig. 5 shows the accuracy comparison between AdaTest and the SVM-based framework of [12], 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. 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. 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. 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. 5. The implementation of a tightly integrated hybrid framework which could optimally incorporate AdaTest and the SVM-based method is currently under development.

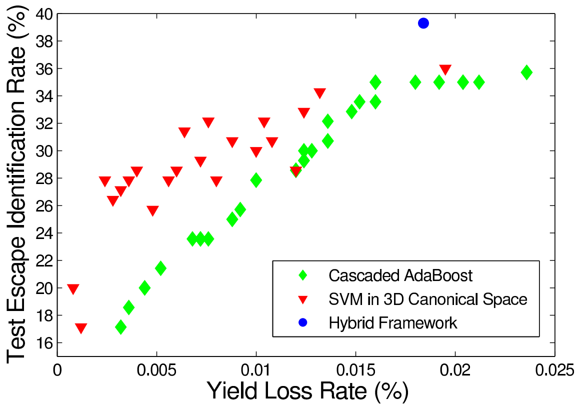
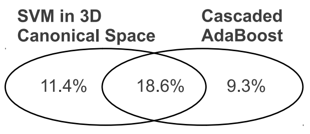
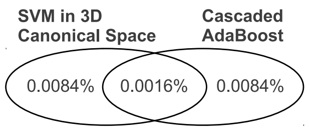


Fig. 5: The ROC curves of classifications based on cascaded AdaBoost and SVM in 3-dimensional canonical space.



(a) Identified test escapes



(b) Yield loss

Fig. 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.

C. 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 application and could limit the real-time application of statistical tests. Table I 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 II. 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 IV-B, 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 I: 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

TABLE II: 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

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. 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 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. 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

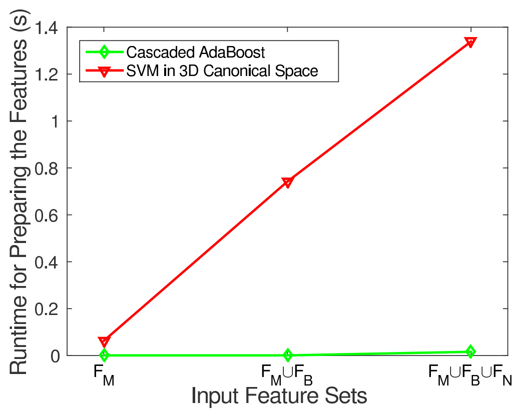


Fig. 7: The runtime for generating and transforming the features before classification.

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.

D. Feature Selection

As mentioned in Section II-A, 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 details of the test items could not be revealed in this paper, the above example demonstrates the information this framework provides for better understanding the characteristics of the test escape population.

V. Conclusions

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 83X and memory usage by at least 3X 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.

Acknowledgments

We would like to acknowledge the Semiconductor Research Corporation (SRC) and the Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan for their support on this work, and the input by Dr. Chun-Lung Hsu of ITRI and Chieh-Chi Kao of UCSB.

References

[1] F. Lin, C.-K. Hsu, and K.-T. Cheng, "Learning from production test data: Correlation exploration and feature engineering," in *IEEE Asian Test Symp. (ATS)*, Nov. 2014.

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

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

[4] 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.

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

[6] 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)*, Sep. 2013.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] F. Lin, C.-K. Hsu, and K.-T. Cheng, “Feature engineering with canonical analysis for effective statistical tests screening test escapes,” in *Proc. Int’l Test Conf. (ITC)*, Oct. 2014.

[13] P. Viola and M. J. Jones, “Robust real-time face detection,” *Int’l Jour. Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[14] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Jour. Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

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

[16] B. Rosner, “Percentage points for a generalized esd many-outlier procedure,” *Technometrics*, vol. 25, no. 2, pp. 165–172, May 1983.

[17] NIST/SEMATECH, “e-handbook of statistical methods,” *http://www.itl.nist.gov/div898/handbook/*, 2003.

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

[19] 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.

[20] 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)*, Sep. 2011.

[21] 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.

[22] 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)*, Sep. 2009.

[23] 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.