An Artificial Neural Network Approach for Screening Test Escapes

Fan Lin and Kwang-Ting Cheng*
Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA
*School of Engineering, Hong Kong University of Science and Technology, Hong Kong

***Abstract*—In this paper we investigate the application of an artificial neural network (ANN) for screening test escapes. Specifically, we propose to train an autoencoder, an ANN, in an unsupervised way to fit the good chip population, i.e. using good chips only as the training set.**

The autoencoder is designed with both its input and output layers representing a set of features that characterize the test data of the chips under test, where we use the Euclidean distance between the values in the input and output layers as the cost function for training. Based on the trained autoencoder, if the test measurement of a query chip has an abnormally large value for the cost function, the chip is likely to be a test escape because it does not fit the characteristics of the good chip population captured by the model.

We demonstrate that an autoencoder-based classification could achieve a higher detection rate for test escapes and a significant reduction in runtime and memory usage, compared with an SVM applied on the same features and some additional proximity features generated from multiple nonlinear transformations.

I. Introduction

Test programs for modern chips contain various test items and cover multiple design/process corners, which results in a significant amount of production test data. Even with the large number of test items applied in multiple environment settings, there are still *test escapes*, which are chips that pass the entire test program but fail later at system-level applications. In addition to the pass/fail results for each test item, there exist multivariate correlations in the test data [1]. By exploring and modeling these hidden correlations, we could learn more about the population of the chips under test and detect anomalies which have a high probability of being test escapes.

Anomaly detection based on semiconductor production test data can be applied at different production test stages. For wafer-level anomaly detection, Sumikawa *et al.* [2] examined the spatial patterns on the wafer, derived a subset of tests in which some novel patterns could be exposed, and searched for wafers with similar abnormal patterns. Zhang *et al.* [3], [4] and Huang *et al.* [5] decomposed the spatial patterns on wafers and clustered the wafers for detecting anomaly wafers. For chip-level anomaly detection, Daasch *et al.* [6]–[9] proposed a concept of *residual*, derived from the measurement values of a chip and those of the neighboring chips to detect outlier chips. O'neill [10] and Sumikawa *et al.* [11] screened outlier chips using principal component analysis (PCA). Krishnan and Kerkhoff [12] proposed outlier screening through exploring multiple Mahalanobis distances. Siatkowski *et al.* [13] proposed an outlier model generalization method in the temporal and spatial domains.

For screening test escapes, Lin *et al.* [14], [15] proposed using multiple types of *residual vectors* based on different expected values as the input features for machine learning algorithms to reveal different aspects of the chips under test. Based on the generated features, they proposed a post-process to derive additional proximity features [16]. The proximity features are calculated using multiple distance/proximity functions between each pair of chips on the same wafer, and the experimental results showed the proximity features could provide additional relevant information about the chip population and improves the classification accuracy for detecting test escapes.

Artificial neural networks (ANNs) have demonstrated great potential and outperformed many other machine learning algorithms in applications such as image and voice recognition. An ANN has 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 paper, we propose using a simplified *autoencoder* [17] structure for classifying test escapes. Using unsupervised learning, we train the autoencoder with good chips only, so that the autoencoder would fit the good chip population. Based on the trained autoencoder, we could then classify a query chip based on how the autoencoder model fits the query chip. For a test escape with some intrinsic defects, the autoencoder model may not fit it as well as the model fits other normal chips.

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 another linear transformation process, called canonical analysis [14], followed by a support vector machine (SVM) classification. It was demonstrated in [14] that canonical analysis could significantly improve the runtime and, in some cases, the accuracy of a classic SVM classifier. In [16], a collection of nonlinear transformations were proposed to generate additional information that further improves the test escape detection rate compared with the framework in [14]. 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 paper, the basic concept of artificial neural networks and the proposed structure will be discussed in Section II and Section III. Section IV illustrates data processing techniques for generating features that characterize the chips under test. Section V presents the experimental results, and Section VI concludes the paper.

II. Artificial Neural Networks

An artificial neural network is composed of multiple neurons. Fig. 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*:

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

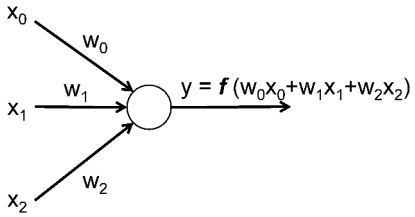


Fig. 1: An artificial neuron with three inputs and one output. The output of a neuron is the activation result of the weighted sum of the neuron's inputs.

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. 2. The structure in Fig. 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. An error based on a cost function is calculated at the output layer and 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.

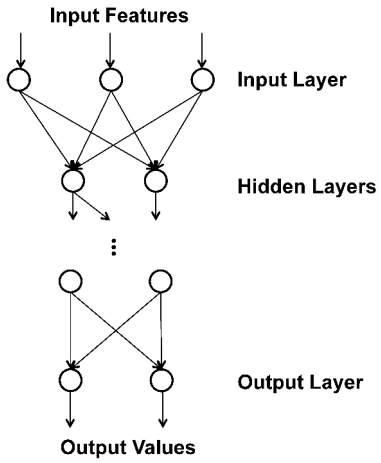


Fig. 2: A neural network contains an input layer, an output layer, and some hidden layers in between.

III. The Proposed Structure

In this paper, we use a specific neural network structure, an *autoencoder* [17], 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 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 at this time. 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 V-B, 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. 3 shows the selected structure, with the following details:

- 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 two conditions:

$$n_{in} = n_{out} \tag{2}$$

$$n_h < n_{in} \tag{3}$$

  For the dataset we analyzed in this paper, $n_{in} = n_{out} > 700$, 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.

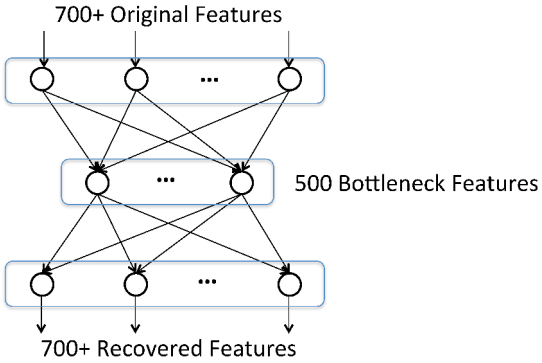


Fig. 3: The proposed autoencoder structure.

We use the Caffe package from UC Berkeley [18] for handling and solving for the neural network model, and use *Adam solver* [19] 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 [20]. Details of the Adam solver can be found in [19].

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 V will demonstrate that this specific structure and cost function could achieve higher classification accuracy than a canonical analysis followed by SVM classification [14] and a collection of nonlinear transformations based on proximity information between each pair of chips on the wafer [16].

IV. Feature Processing

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

A. 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 [15]. 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:

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

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 [21].

B. Feature Generation

In this analysis, we use the *residual vectors* proposed in [14] 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 $\mathbf{r}$:

$$\mathbf{r} = \mathbf{x_m} - \mathbf{x_e} \tag{5}$$

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

Defined as the difference between the measurement values and the 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 [15] 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 [22] spatial pattern of the wafer, and 3) the median of the eight closest neighbors' measurements of the query chip. In [14] and [15], analysis and experimental data have shown that each type of the residual vectors could potentially reveal a unique subset of the test escapes that the other types of the residual vectors could not.

C. Proposed Test Flow

Fig. 4 shows the proposed flow of using the autoencoder to generate the features and classify test escapes, based on the parametric measurements in production test data. Although only the good chips in the training set are used for training the autoencoder, both the good chip population and test escape population are used for deriving a threshold on the Euclidean distance between the input and output values of the autoencoder. During test application, the Euclidean distance between the values in the input and output layers in the trained autoencoder is calculated for each query chip, and a query chip with a Euclidean distance greater than the threshold will be classified as a test escape.

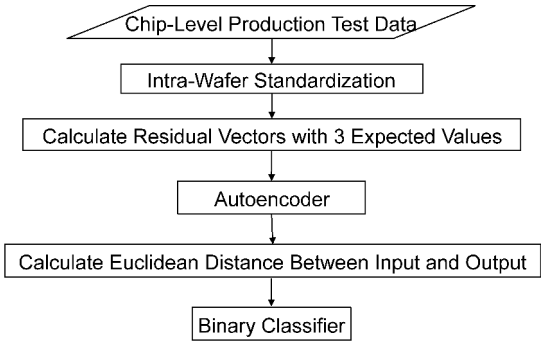


Fig. 4: The flow of using the proposed autoencoder for test escape screening.

V. 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 [23] (for the comparison described in Section V-C), and the rest 200+ wafers as the testing set.

A. Test Escape Emulation

Since this dataset does not contain actual test escape information, we first *emulated* a test escape population for our analysis using intrinsically defective chips with minor failures. Based on production test data, we selected chips that failed only one parametric test item and replaced the value that falls out of the limits of the test spec with the median measurement value of the good chip population of the corresponding test item. The measurement values of all other test items are kept intact. With such manipulation, these selected chips would pass the entire test program, while still having the subtle syndromes in the all-but-one test items. The goal for our analysis is then to detect the abnormal signals in the non-failing test measurements of the *emulated test escapes*.

To create a more diverse test escape population with a practical number of parts per million (PPM), we further selected test items

that hiding each of them would increase the emulated test escape population by more than 50PPM, and removed the emulated test escapes that were created by hiding these test items from the test escape pool. Therefore, we ensured the emulated test escape population has a higher relative diversity in terms of the root cause (the single test item that a chip failed) of the test escapes. The resulting test escape population corresponds to approximately 560PPM for the testing set.

B. 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* [17], which allows the training process to converge to a good solution faster without searching slowly around some local optima.

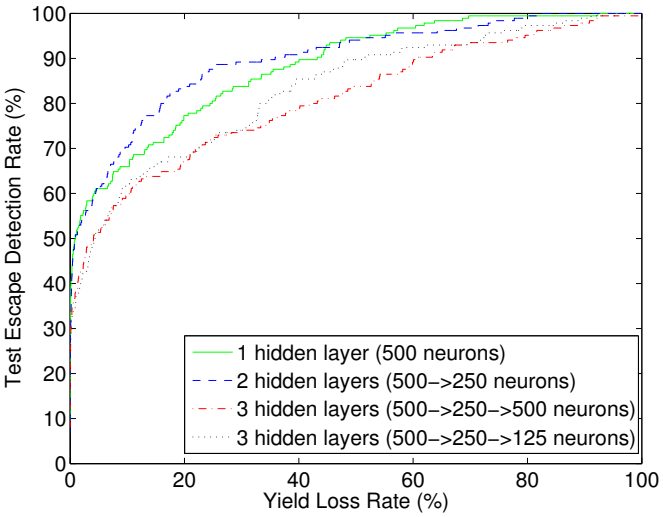
Fig. 5 shows the relative operating characteristics (ROC) curves, which plot the true positive rate (test escape detection rate) versus the false positive rate (yield loss rate), for autoencoder structures with 1) a single hidden layer with 500 neurons, 2) a hidden layer with 500 neurons followed by 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. 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.

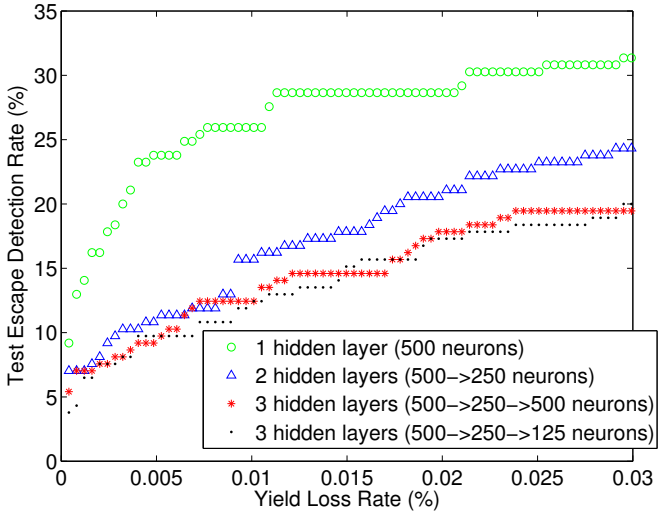
Fig. 5b plots the same ROC curves in the region with very low yield loss rate, which is usually required for the application of test escape screening. Given the very low yield loss rate, the structure with a single hidden layer significantly outperforms the other structures. Although a structure with 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 has modeled the most critical characteristics of the good chip population that could be used to distinguish test escapes in the target region of the yield loss rate.

C. Classification Accuracy

Fig. 6 shows the ROC curves of three frameworks for comparison. In the first framework [14], shown in green triangles, the three



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



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

Fig. 5: The ROC curves demonstrate the classification accuracy for different structure designs of the autoencoder.

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 compacts the separation between classes in the high-dimensional feature space into the first few dimensions in the transformed feature space. It has been demonstrated that applying canonical analysis before SVM for feature reduction can significantly improve the runtime and in some cases the accuracy for classifying test escapes [14]. In the second framework [15], 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 [15]. The classification accuracy of the proposed autoencoder framework is marked by the red dots.

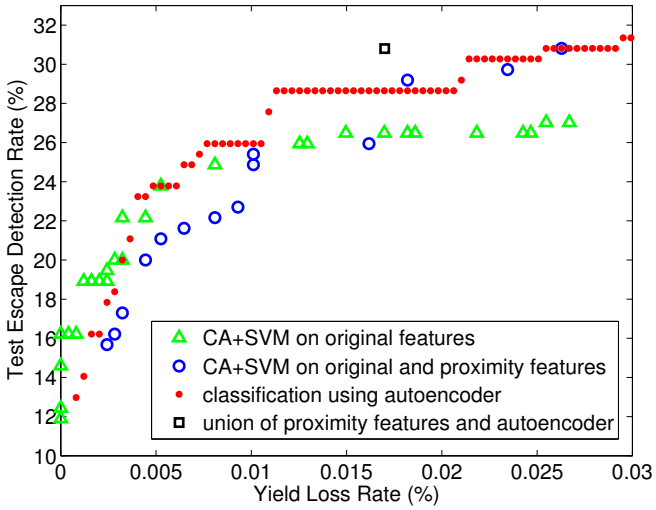
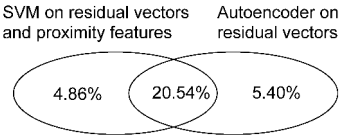


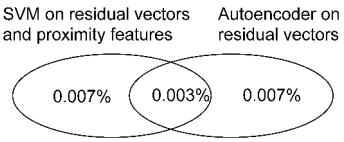
Fig. 6: The ROC curves of three frameworks.

As discussed in [15], 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 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, the classification using autoencoder consistently detects more test escapes when the yield loss rate is below 0.018% (e.g. the autoencoder detects additional 3% test escapes at a yield loss rate of 0.016%). The detection rates of the two classifications become similar at a greater yield loss rate.

To further compare the populations of the detected test escapes by the different frameworks, we plot the Venn diagram of the chips that are identified as test escapes at a yield loss rate of 0.01%, where the second framework and the autoencoder detect similar amount of test escapes, in Fig. 7. Although the autoencoder reveals 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 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 as a naive integration, we could achieve a test escape detection rate of 30.8% at a yield loss rate of 0.017%, which is at least 2% better than either of the two methods alone, as marked by the black square in Fig. 6.



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



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

Fig. 7: The Venn diagrams of the test escape and yield loss populations resulted from the SVM on proximity features and residual vectors (the method of [15]) and from the proposed autoencoder.

D. 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. The absolute values of the trained weights in the hidden layer are demonstrated in Fig. 8 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). 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. 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, shows that these neurons in the hidden layers are relatively less important than the others. 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.

E. 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, and the second framework takes 4.6 seconds per wafer for generating the proximity features and classification. With the use of an NVIDIA GTX 980 graphic card for acceleration, the autoencoder takes 0.1 seconds for the classification per wafer. Compared with the second framework in which a collection of nonlinear transformations are applied, which incurs a significant amount of runtime and memory usage, the linear classification using the autoencoder could reduce the runtime by 46X and achieve a higher classification accuracy.

VI. Conclusions

In this paper, we propose an autoencoder structure that could classify test escapes more accurately than the state-of-the-art statistical and machine learning approaches reported in [14] and [15]. The specific structure and the cost function of the autoencoder, though

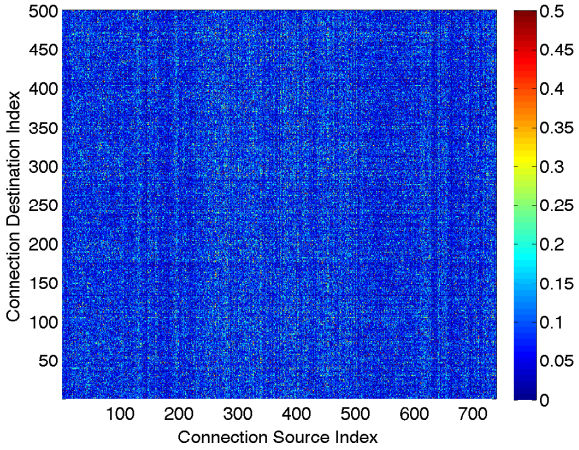


Fig. 8: The absolute values of the weights in the hidden layer as a color-coded map.

only a linear transformation, could reveal even more test escapes compared with a framework incorporating a collection of nonlinear transformations in [15].

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 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 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.

ACKNOWLEDGMENTS

We would like to acknowledge the Semiconductor Research Corporation (SRC) for their support on this work, and Chieh-Chi Kao from UCSB for his input.

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] 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)*, Sep. 2013.

[3] W. Zhang, X. Li, S. Saxena, A. Strojwas, and R. Rutenbar, "Automatic clustering of wafer spatial signatures," in *Proc. IEEE/ACM Design Automation Conf. (DAC)*, May 2013.

[4] W. Zhang, K. Balakrishnan, X. Li, D. Boning, S. Saxena, A. Strojwas, and R. Rutenbar, "Efficient variation decomposition via robust sparse regression," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 1072–1085, Jul. 2013.

[5] K. Huang, N. Kupp, J. M. C. Jr, and Y. Makris, "Process monitoring through wafer-level spatial variation decomposition," in *Proc. Int'l Test Conf. (ITC)*, Sep. 2013.

[6] 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.

[7] 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.

[8] 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.

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

[11] 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.

[12] 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.

[13] S. Siatkowski, C.-L. Chang, L.-C. Wang, N. Sumikawa, L. Winemberg, and W. R. Daasch, "Generalization of an outlier model into a global perspective," in *Proc. Int'l Test Conf. (ITC)*, Oct. 2015.

[14] 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.

[15] F. Lin, C.-K. Hsu, and K.-T. Cheng, "Adatest: An efficient statistical test framework for test escape screening," in *Proc. Int'l Test Conf. (ITC)*, Oct. 2015.

[16] F. Lin, C.-K. Hsu, A. G. Busetto, and K.-T. Cheng, "Pairwise proximity-based features for test escape screening," in *Proc. IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD)*, Nov. 2015.

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

[18] 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.

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

[20] L. Bottou, "Stochastic gradient descent tricks," *Neural Networks: Tricks of the Trade*, 2012.

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

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

[23] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. on Intelligent System and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.