Wafer Pattern Recognition Using Tucker Decomposition

Ahmed Wahba, Li-C. Wang, Zheng Zhang
UC Santa Barbara

Nik Sumikawa
NXP Semiconductors

***Abstract*— In production test data analytics, it is often that an analysis involves the recognition of a conceptual pattern on a wafer map. A wafer pattern may hint a particular issue in the production by itself or guide the analysis into a certain direction. In this work, we introduce a novel approach to recognize patterns on a wafer map of pass/fail locations. Our approach utilizes Tucker decomposition to find projection matrices that are able to project a wafer pattern represented by a small set of training samples into a nearly-diagonal matrix. Properties of such a matrix are utilized to recognize wafers with a similar pattern. Also included in our approach is a novel method to select the wafer samples that are more suitable to be used together to represent a conceptual pattern in view of the proposed approach.**

I. Introduction

Machine learning techniques have been utilized in production yield optimization and other applications in design automation and test in recent years (e.g. [1]). In the context of production yield, the recent work in [2] proposes using machine learning for *concept recognition*. The idea is to view an analytic process in terms of three components as depicted in Fig. 1. The top two components in the analyst layer are driven by domain knowledge. The tool component includes data processing and analytic tools that can be invoked.

In view of the figure, an analytic workflow can be thought of as a software script written by an engineer to execute the analytic process. This script calls a tool when needed. In such a script, a decision node may involve a *concept* perceived by the engineer. Concept classes discussed in [2] include those based on a set of wafer patterns. For example, a wafer pattern captures the concept of "edge failures" (EF). Then, in the script the engineer may write something like "For each wafer map, if (observe EF on the map), do something." In order to automate such a statement, one needs a *concept recognizer* to recognize the wafer pattern.

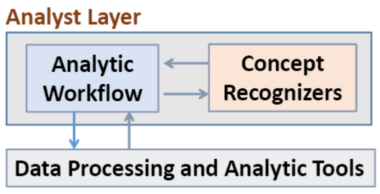


Fig. 1. Three Components to Capture an Analytic Process

The work in [2] employs the approach of Generative Adversarial Networks (GANs) [3][4] for building a concept recognizer. As pointed out in [2], it is well known that such a deep neural network model can have *adversarial examples* [5], a slightly perturbed input sample that causes the sample to be misclassified. This "inherent" issue for using a deep neural network motivates the search for a second approach to implement a concept recognizer.

Recent advances in tensor analysis [6] provide an opportunity for developing an alternative approach for concept recognition. However, the concept recognition proposed in [2] is unsupervised learning. Tensor analysis techniques are usually for general data processing (e.g. compression), and not developed specifically as a technique for learning or unsupervised learning. Although it has been used in the context of machine learning, for the purposes such as compressing neural network layers [7] and speeding up the training with a deep neural network [8], it has not been widely used as a stand-alone learning technique. Therefore, it would be interesting to investigate if a Tensor analysis technique can be turned into an approach for concept recognition.

This work considers concepts only in terms of wafer patterns. For training, a wafer pattern concept is represented with a small set of wafer maps each plotting the locations of passing and failing dies in terms of two distinct colors. Once a model is learned, it can be used to check if a given wafer map falls within the respective concept.

There are two main contributions in this work: 1) This work shows that it is feasible to use tensor analysis, specifically Tucker decomposition, to implement a wafer pattern concept recognizer. 2) The tensor based approach provides a way to implement a *learnability* measure. The importance of such a measure is that it can be used to decide if a given set of training samples is suitable for learning a concept. If not, it can be used to select a subset for training a better model, effectively taking a given concept and refining it.

Note that the scope of the work is to provide a feasible alternative to the GANs-based approach proposed in [2] for wafer pattern concept recognition. The proposed tensor-based approach is not meant to be a replacement. A study of the two approaches combined is not included in this work. This is because such a study should not be as simply as showing the results based on a list of wafer patterns to be recognized. Tensor analysis can be integrated into neural network based learning [7][8] to provide a hybrid approach. As a first step, this work treats tensor analysis as a standalone technique for concept recognition and focuses on understanding its feasibility.

II. Tensor Notations

Tensors are the generalized form of vectors and matrices. While vectors are one dimensional, matrices have two dimensions, tensors can have more than two dimensions.

For example, Fig. 2 shows a three-way tensor $\boldsymbol{\mathcal{A}}$ and its Tucker decomposition, which will be explained later. An n-way tensor has n dimensions or "modes".

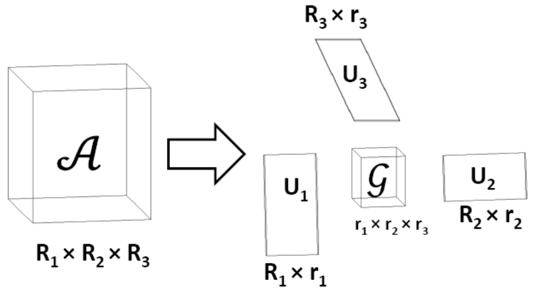


Fig. 2. Tucker Decomposition for a Three Dimensional Tensor

Tensor analysis and decomposition has been used in many applications which involve dealing with large amounts of data. In [6], the authors discuss the main tensor decompositions and their applications. The work in [7] uses tensors to represent a deep neural network and shows huge memory savings while maintaining the same prediction accuracy. The work in [8] represents the convolution layers of a deep neural network and show significant speedup in the training time while maintaining less than 1% loss in accuracy. Tensor analysis can also be used for dimensionality reduction [9] and low rank approximation [10], [11].

In this paper, we will follow the notations used in [6]. A tensor is represented by a boldface Euler script letters, for example $\boldsymbol{\mathcal{A}}$. A matrix is represented by a boldface uppercase letter, for example $\mathbf{A}$. And a vector is represented by a boldface lowercase letter, for example $\mathbf{a}$. In the following, we introduce some basic tensor operations needed in this work.

A. Mode Multiplication

Mode-i Multiplication is the operation of multiplying a tensor and a matrix along a mode-i, and is denoted by the operation $\times_i$. The second dimension of the matrix has to agree with the size of mode-i of the tensor. The result is another tensor with the size of mode-i replaced by the size of the matrix first dimension. For example, $\boldsymbol{\mathcal{A}}_{7\times5\times3} \times_3 \mathbf{A}_{6\times3} = \boldsymbol{\mathcal{B}}_{7\times5\times6}$.

B. Tucker Decomposition

Tucker decomposition [11] is the high dimensional generalization of matrix Singular Value Decomposition (SVD) [12]. As shown in Fig. 2, in Tucker decomposition, a n-dimensional tensor $\boldsymbol{\mathcal{A}}$ (e.g. $n = 3$) of size $R_1 \times R_2 \times R_3$ is decomposed into n+1 components: a core tensor $\boldsymbol{\mathcal{G}}$ of size $r_1 \times r_2 \times r_3$ and n orthogonal projection matrices $U_{i\,i=1}^{\,n}$ each of size $R_i \times r_i$ such that $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}} \prod_{i=1}^{n} \times_i U_i$. The choice of the core size, or what is referred to as the intermediate ranks $r_1, r_2, ..., r_n$ determines how accurate the decomposition is in representing the tensor $\boldsymbol{\mathcal{A}}$. The closer $r_1, r_2, ..., r_n$ to $R_1, R_2, ..., R_n$ the higher the accuracy. For details about Tucker decomposition please refer to [6].

In this work, for building a recognizer model, we would desire the accuracy to be as high as possible in representing a tensor. Consequently, we chose $r_1, r_2, ..., r_n$ to be equal to $R_1, R_2, ..., R_n$. Our implementation of Tucker decomposition involves getting an approximate decomposition using Higher Order Singular Value Decomposition (HOSVD) [13] and using this approximation as a starting point in the iterative Higher Order Orthogonal Iteration (HOOI) [14].

III. Wafer Pattern Recognition

Given a set of wafer maps as the training samples to learn a wafer pattern concept, each wafer essentially is a matrix, where each die is an element of this matrix. Failing dies are indicated by a value "-1", passing dies are indicated by a value of "+1" while "0" indicates no die present at the location. Fig. 4 (discussed in the next section) shows some examples of wafer maps. They represent two distinct concepts: an Edge Failures concept on the first row, and a Systematic (grid) Failures concept in the second row.

To build a recognizer model, the training wafer maps are put together to form a 3-dimensional tensor. Tucker decomposition is then applied to the tensor to get a core tensor, along with the three projection matrices. Since the focus is on capturing the pattern on each wafer, the projection matrix along the third mode U_3 is discarded. This is because this projection matrix records the information regarding to wafer-to-wafer variations.

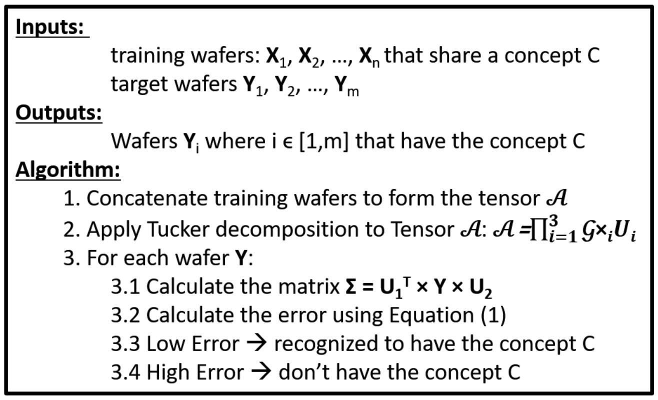
The first two projection matrices are used to produce transformed wafer maps for each original wafer map, by multiplying the original wafer map with the first two projection matrices, $\Sigma_p = U_1^T \times X_p \times U_2$ where X_p is a matrix representing the original wafer map. In our experiments, we observed that such transformed matrices are very close to being diagonal (if the given wafer maps are very similar). In fact, if the wafer maps are exactly the same, the transformed matrices Σ_i are exactly diagonal. It's also worth mentioning that in that case, the $X_p = U_1 \times \Sigma_p \times U_2^T$ is the Singular Value Decomposition of the wafer map X_p.

We also observed that if a wafer map X_q has a different pattern than the training wafer maps, the resulting transformed matrix Σ_q is neither diagonal, nor close to being diagonal. Hence, in order to recognize the wafer maps with a similar pattern as the training wafer maps, we can use the first two projection matrices obtained from the Tucker decomposition to get a transformed matrix Σ and check how "diagonal" it is.

In order to do this, we need a measure of "diagonal-ness." Our measure is based on the following error equation:

$$Error = \frac{\sum \sigma_{i \neq j}^{ij\;2}}{\sum \sigma_{ij}^2{}_{i=j}} \tag{1}$$

Where σ_{ij} is the element on row i and column j in the transformed matrix Σ. The lower the error, the more diagonal the matrix is. The algorithm is summarized in Fig. 3.



```
Inputs:
        training wafers: $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n$ that share a concept C
        target wafers $\mathbf{Y}_1, \mathbf{Y}_2, \ldots, \mathbf{Y}_m$
Outputs:
        Wafers $\mathbf{Y}_i$ where i ϵ [1,m] that have the concept C
Algorithm:
    1. Concatenate training wafers to form the tensor $\mathcal{A}$
    2. Apply Tucker decomposition to Tensor $\mathcal{A}$: $\mathcal{A} = \prod_{i=1}^{3} \mathcal{G} \times_i U_i$
    3. For each wafer Y:
        3.1 Calculate the matrix $\mathbf{\Sigma} = \mathbf{U}_1^T \times \mathbf{Y} \times \mathbf{U}_2$
        3.2 Calculate the error using Equation (1)
        3.3 Low Error → recognized to have the concept C
        3.4 High Error → don't have the concept C
```

Fig. 3. Algorithm 1: Tensor-Based Concept Recognition

IV. Building a Collection of Recognizers

To show how the proposed algorithm is used for concept recognition, wafers from two different concepts are illustrated in Fig. 4. The first concept can be called the 'Edge Failures' and the second concept the 'Systematic Failures'.

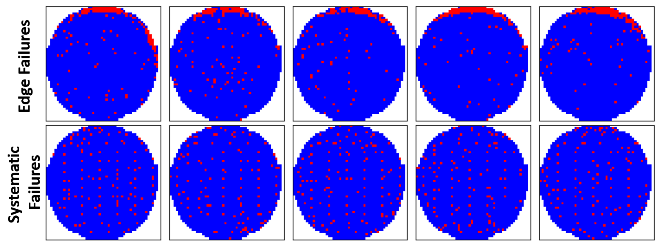


Fig. 4. Sample Wafer Maps From the Two Concepts

Then Fig. 5 shows five wafer maps used as the training samples to learn the first concept using the proposed algorithm. The recognizer is then applied to scan 8300 wafer maps collected from an automotive SoC product line and the sorted error values are shown in Fig. 6.

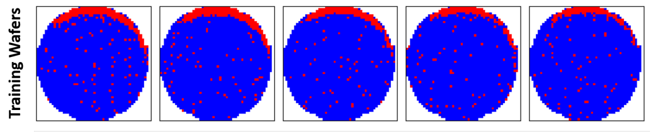


Fig. 5. Training Wafer Maps for Edge Failures

In Fig. 6, wafer maps with Edge Failures are marked. They are manually picked and there are 35 of them. It is clear from the figure that there are some wafer maps with substantially large error values on the right. However, the wafer maps of Edge Failures have error values, while small, spread across a large range that also contains many other wafer maps without Edge Failures. This indicates that a direct application of the proposed algorithm would not work.

Suppose we focus on a small set of wafer maps. In this case, we use 15 wafer maps from the first concept and 15 wafer maps from the second concept in Fig. 4. We apply the recognizer to this small set of 30 wafer maps. Fig. 7 plots

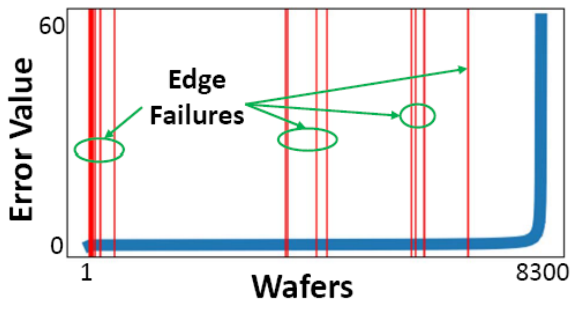


Fig. 6. Error Value for All 8300 Wafers

the resulting error values after sorting. In this figure, it is clear that the two types of wafer maps can be separated by setting a threshold around 0.4.

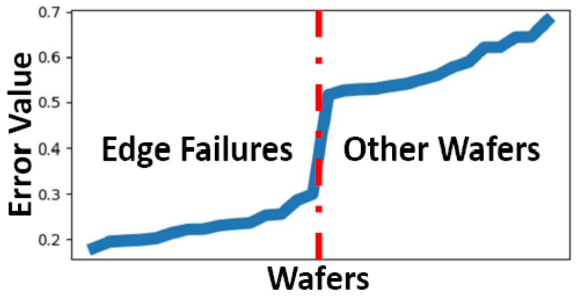


Fig. 7. Error Value for The Two Concepts

A. Multi-Concept Recognizer Set

The two experiments above motivate us to follow an approach that treat concept recognizers as a collection. The idea is that given a wafer map, this set of recognizers are applied in parallel and the wafer map would be recognized as the concept by the recognizer reporting the smallest error value. This approach has the advantage that there is no need to set a threshold as the algorithm in Fig. 3 might have suggested, hence removing the subjectivity in concept recognition with the algorithm.

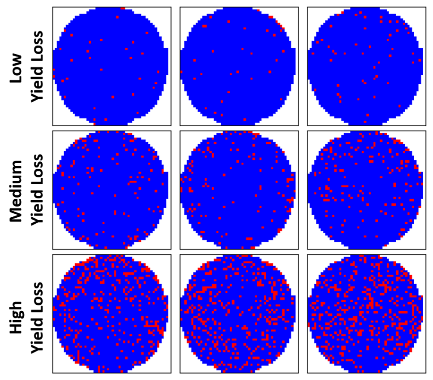


Fig. 8. Different Failure Densities for Normal Failure Wafers

However, to implement the multi-recognizer approach, we will need a recognizer for the concept of a 'Normal' wafer map. This Normal concept recognizer can then serve as the baseline to other concept recognizers.

To implement the approach, we first categorize all wafers into three groups (Low Yield Loss, Medium Yield Loss, and High Yield Loss) based on their yield loss. Fig. 8 shows some examples in each group. The low-loss group contains about 70% of the wafers. For those wafers, they are not of interest. For the medium-loss and high-loss groups, a concept recognition set is developed for each group.

B. *Medium Yield Loss Wafers*

There are about 2K Medium yield loss wafers. In addition to the Normal concept, three other concepts are identified: Edge Failures, Systematic Failures, and Center Failures. They are illustrated in Fig. 9.

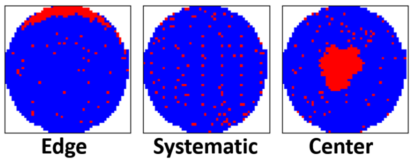


Fig. 9. Concepts Identified Within the Medium Yield Loss Wafers

For each concept, five wafer maps are used to train its recognizer. In total there are four recognizers in this group. The set of recognizers are applied to scan the 2K wafer maps. Examples of recognized wafer maps for the three concepts are shown in Fig. 10.

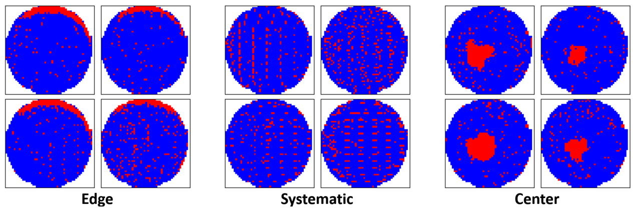


Fig. 10. Examples of Recognized Wafer Maps

C. *High Yield Loss Wafers*

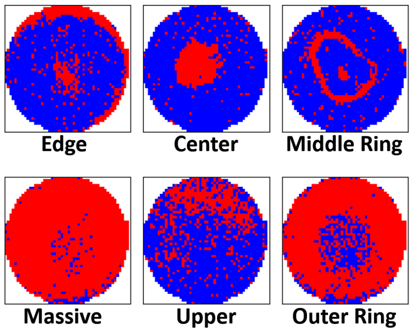


Fig. 11. Concepts Identified Within the High Yield Loss Wafers

In this group, six concepts (Edge Failure, Center Failure, Middle Ring Failure, Massive Failure, Upper Failure, Outer Ring Failure) are identified in addition to the Normal concept. Fig. 11 illustrates these six concepts.

Three training wafer maps are chosen as the training samples for each concept. Examples of recognized wafer maps for each of the six concepts are shown in Fig. 12.

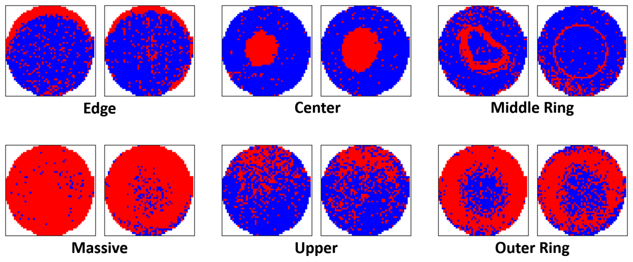


Fig. 12. Recognized Wafer Maps in Each of the Six Concepts

1) An observed issue: Fig. 13 shows three wafer maps that are misclassified by the collection of the seven recognizers. These wafer maps show that they should have been classified into the Center Failures concept, but the collection recognizes them as wafers of Outer Ring Failure.

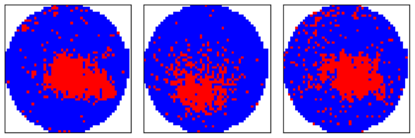


Fig. 13. Center Failures Recognized as Outer Rings

The mistake had something to do with how a wafer map is represented. For a wafer map, passing parts and failing parts are denoted as +1 and -1 values, respectively. As a result, the matrix representing a wafer map of Outer Ring Failure and the matrix representing a wafer of Center Failure can have opposite values on most of the entries. In other words, if we flip the two values, they are similar. Consequently, their transformed matrices are similar except that all signs are flipped. The error value calculation used earlier does not take this into account, resulting in the misclassification seen. To avoid this issue, a sanity check on the sign of the largest diagonal value of the transformed matrix is added. This simple modification resolves the issue and enables correct classification between the two concepts.

D. *An Observed Limitation*

A major limitation was observed with the proposed Tensor-based approach for concept recognition when compared to the GANs-based approach proposed in [2]. The results in [2] show that a GANs-based recognizer can be trained to recognize a pattern even though the pattern is rotated. This is especially the case for Edge Failures pattern which may appear in different direction on a wafer map. In [2], a recognizer is trained to recognize edge failures regardless of where the failures concentrate on. On the other hand, we were unable to produce a single recognizer to do so using the proposed approach. Our recognizer only recognize edge failures in the same direction as that shown on the training samples. For example, Fig. 14 show several examples not recognized by the Edge Failures recognizer from the medium-loss group (or the high-loss group).

In [2], a single recognizer was built by taking five training samples all with edge failures in a similar direction and

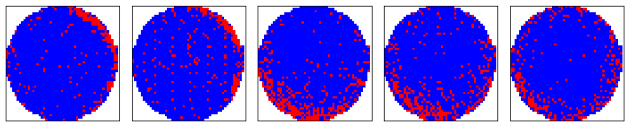


Fig. 14. Rotated Edge Patterns that are Missed by Our Concept Recognizer

rotating each clockwise to produce 11 other maps. This gives 60 wafer maps in total for training. When this method is used with our approach, the result is that the recognizer would become basically a recognizer for the low-yield-loss concept, i.e. wafer map containing almost no failure. This is because Tucker decomposition is not rotation-invariant. Hence, when an edge pattern appear in all 12 directions, they are treated as "noise" in the data. Their "commonality" essentially is an empty wafer map with no failure.

V. Improving Training Set

One of the challenges in practice for building a concept recognizer is choosing the training samples. This is because concept recognition is based on unsupervised learning. When a set of training samples are given, it is unknown if the training samples should be treated as a single concept class or multiple concept classes. It would be desirable to have a way to assess that question.

The $Error$ in equation (1) provides a convenient way to develop a method for that purpose. The quantity $LB = 1 - Error$ can be thought of as how good the projection matrices from the Tucker decomposition can be used to represent a wafer map, i.e. similar to the notion of model *fitting* in machine learning. For example if the Tucker model is built from a set of identical wafer maps, we would have $LB = 1$ for every wafer. Intuitively, we can use LB to indicate how well a tucker model fits a wafer map.

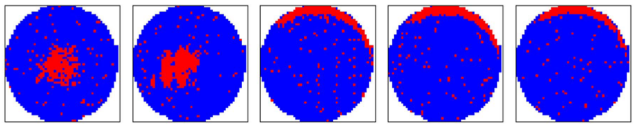


Fig. 15. A Training Set With Samples From Two Concepts

To illustrate the point, Fig. 15 shows five wafer maps from the Edge Failures and Center Failures classes in the medium-loss group. After Algorithm 1 is applied to these five samples, their LB values are [0.24, 0.19, 0.58, 0.62, 0.67] following the same order as shown in the figure. Observe that the LB values for the two Center Failures wafer maps are noticeably lower than the other three.

Training Wafers	Learnability Vector	Average
Edge Failures	[0.76, 0.77, 0.81, 0.77, 0.74]	0.77
Systematic Failures	[0.77, 0.77, 0.74, 0.75, 0.76]	0.76
Center Failures	[0.61 , 0.58 , 0.61 , 0.47 , 0.35]	0.52
Normal Failures	[0.74, 0.74, 0.73, 0.73, 0.72]	0.73

TABLE I

LB values from the medium-loss group

Recall that for the concepts in the medium-loss group, five samples are used in training. Table I shows their LB values after the training. Observe that the average LB value in the Center Failures case is much lower than others while the last two samples have noticeably lower LB values. This indicates that for this concept class, there might be room for improvement in terms of choosing a better training set.

A. Learnability Measure

Our method to improve a training set is based on calculating the average LB value. Suppose we are given a set S of n samples. The goal is to choose a subset of samples as our training set. Suppose we have also a test set T of m samples. Samples in S and T are manually selected and visually determined to be in the same concept class. The goal is to choose the best set of samples from S to build a Tucker model. Note that from this perspective, it might be more intuitive to think the method is for filtering out "noisy" samples rather than for "choosing" samples.

The idea is simple. Suppose samples in S are ordered by their LB values as $s_1, \ldots, s_n$. Let $S_i = \{s_1, \ldots, s_i\}$. The average LB_i is calculated by applying the Tucker model from the set S_i to T. For example, let $n = 10$ and $m = 15$. Fig. 16 shows the average LB results for the four concept classes from the medium-loss group.

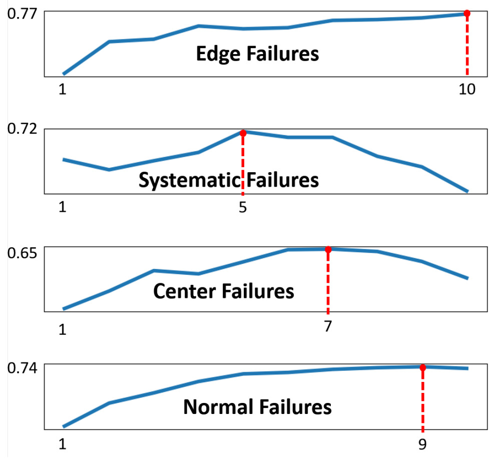


Fig. 16. Deciding the Best Training Set Using Average LB

As shown in the figure (x-axis is the number of wafer maps and y-axis is the average LB), for Edge Failures, it reaches the best result when all 10 training wafers are used. For others, the best result happens with fewer samples. The models were re-built using the new training sets. Fig. 17 shows some examples of wafer maps previously misclassified and now correctly classified by the new recognizers. This shows that the learnability measure enables the use of a better training set, resulting in more accurate recognition. Notice that for the Systematic Failures concept, the number of training samples is five, same as before.

VI. A Final Remark

The intuition behind Tucker decomposition and the intuition behind Principle Component Analysis (PCA) share some similarity. Because PCA can be used as an outlier detection technique (see e.g. [15] for general discussion and

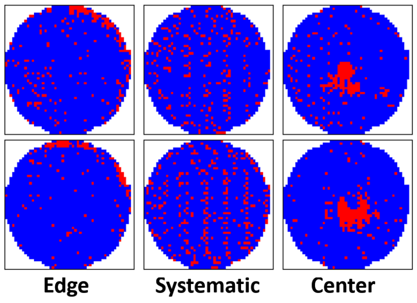


Fig. 17. Examples of Recognized Wafers After Training Using the Improved Training Sets

[16] for its use in outlier analytics with test data), it would be intuitive to think that Tucker decomposition can be used in the context of outlier analysis as well. In fact, this is indeed the case. For example, the work in [17] presents a technique called Dynamic Tensor Analysis (DTA), which conceptually is an approximated Tucker decomposition, and uses DTA as an outlier detection method to capture anomalies in network traffic data.

In DTA, projection matrices are calculated for a first matrix, and are updated (refined) for every subsequent matrix incrementally. In each incremental step, the goal is to minimize the average reconstruction error (e) for all matrices seen so far, where $e = \sum_{t=1}^{n} \left\| \mathcal{X}_t - \mathcal{X}_t \prod_{i=1}^{M} \times_i (U_i U_i^T) \right\|^2$.

If one views wafer production as providing a stream of data, then it seems that DTA can also be used to detect abnormal wafers, for example based on their wafer maps. We implemented this idea and applied DTA with the 8300 wafers. Fig. 18 shows some of the inliers and some of the outliers while Fig. 19 shows where their outlier scores stand in the rankings of the 8300 wafers.

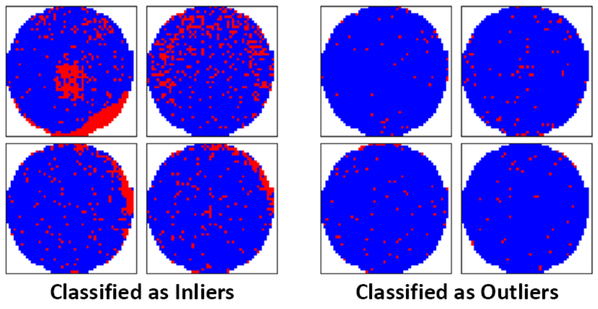


Fig. 18. Issue With Using DTA in Wafer Outlier Detection

As seen, DTA's inliers/outliers are not consistent with our intuitive perception for what an outlier/inlier should be. Of course, most of the DTA's inliers/outliers are not as bad as those shown in Fig. 18. However, those examples do illustrate the challenge to use DTA for detecting outlier wafer maps. One can say that the view to define an outlier by DTA is not consistent with the view to perceive an outlier by a person, but making these two views consistent can be challenging. This was the reason why we used Tensor analysis in concept recognition and not in outlier detection, even though both can be thought of as a form of unsupervised learning.

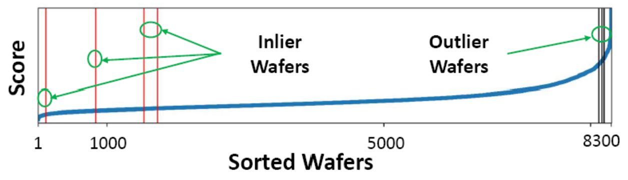


Fig. 19. Sorted Wafers According to Outlier Score

VII. Conclusion

In this work, Tucker decomposition is employed to build models for recognizing concepts appearing on wafer maps. A learnability measure is proposed to select training samples and the effectiveness of various concept recognizers are illustrated. Future work includes a deeper understanding of the strengths and weaknesses between the proposed approach and the GANs-based approach proposed in [2] for concept recognition. It will also be interesting to investigate a hybrid approach combining the two.

References

[1] L.-C. Wang, "Experience of data analytics in EDA and test - principles, promises, and challenges," *IEEE Trans. on CAD*, vol. 36, no. 6, pp. 885–898, 2017.

[2] M. Nero, J. Shan, L. Wang, and N. Sumikawa, "Concept recognition in production yield data analytics," in *IEEE International Test Conference*, 2018.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and J. Bengio, "Generative adversarial networks." arXiv:1406.2661, 2014.

[4] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs." arXiv:1606.03498v1, 2016.

[5] C. Szegedy and et al., "Intriguing properties of neural networks." arXiv:1312.6199v4, 2013.

[6] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

[7] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems*, pp. 442–450, 2015.

[8] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[9] L. Kuang, F. Hao, L. T. Yang, M. Lin, C. Luo, and G. Min, "A tensor-based approach for big data representation and dimensionality reduction," *IEEE transactions on emerging topics in computing*, vol. 2, no. 3, pp. 280–291, 2014.

[10] R. A. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis," 1970.

[11] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[12] V. Klema and A. Laub, "The singular value decomposition: Its computation and some applications," *IEEE Transactions on automatic control*, vol. 25, no. 2, pp. 164–176, 1980.

[13] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[14] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[15] I. Jolliffe, *Principal Component Analysis*. Springer, 1986.

[16] P. M. O'Neil, "Production multivariate outlier detection using principal components," in *International Test Conferenceh*, IEEE, 2008.

[17] J. Sun, D. Tao, and C. Faloutsos, "Beyond streams and graphs: dynamic tensor analysis," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 374–383, ACM, 2006.