# A Deep Learning Model for Redundancy Analysis Algorithm Recommendation

Atishay Kumar[1], Helik Kanti Thacker[1], Ankit Gupta[1], Keerthi Kiran Jagannathachar[1], Deokgu Yoon[2]
*DRAM Solutions[1] , DRAM Product Engineering Team[2]*
*Samsung Semiconductor India Research and Development[1] , Samsung Electronics[2]*
{atishay.1, h.thacker, ankit.g2, keerthi.k, deokgu.yoon}@samsung.com

***Abstract*—Manufacturing errors, external impurities or faulty deposition during chip fabrication could generate chips with faulty memory cells, rendering the chip unusable. To repair these faulty memory cells, redundancies are included in the memory in the form of spare rows and columns. The process of mapping faulty lines to redundant cells is Redundancy Analysis. Applying a uniform Redundancy Analysis algorithm on the wafers or running algorithms sequentially one after the other would either compromise on the repair time or wafer yield. An end-to-end solution for memory repair is proposed in this paper. A clustering algorithm to classify, identify and extract features from chip errors on a wafer is proposed. These features along with other derived parameters are used as an input to the neural network recommender system to select algorithms allowing an increase in the wafer yield keeping a low repair time per wafer. We have performed comparisons of the generated result with and without clustering and with other methods of classification of chips for Redundancy Analysis algorithm selection such as Decision Trees. Experimental results demonstrate that this solution out-performs the heuristic algorithmic solutions by 9.1% and 32.9% in terms of yield for medium and high error rates.**

***Keywords— Deep Learning, Clustering, Memory Repair, DRAM, Redundancy Analysis***

## I. INTRODUCTION

Manufacturers have been able to keep up with the increasing semiconductor demands by producing chips in large quantities with higher densities on a single wafer. The increase in wafer density coupled with the decrease in node sizes have led to an increased probability of chip defects, which reduces the wafer yield. Beginning with 64Kbit DRAM chips, manufacturers have included redundancies (spare rows and columns) in the chips to repair them which improves the overall wafer yield. Using wafer tests during the Electrical Die sorting process (EDS), the defect addresses on each chip can be located. We use these addresses to perform Redundancy Analysis (RA), which is a process of allocating spare rows and columns to the defective addresses detected in the chip, as illustrated in Figure 1 where row 2 and columns 4 and 6 are repaired using the spares available.

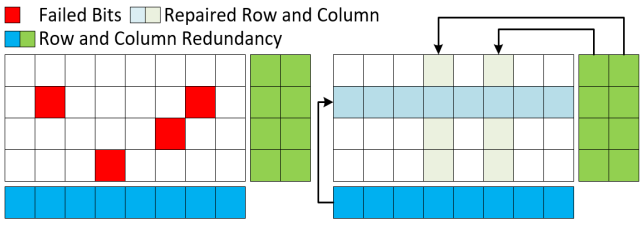


Fig. 1: Memory repair using redundant rows and columns

RA is an NP-complete problem [1], so with an increasing number of chip errors, the time required to repair the chip with maximum possible yield increases exponentially. RA allows the repair of defective chips, but if the chip is unrepairable due to a large number of errors, it would be discarded. Different RA Algorithms like Broadside [2], LECA [3] and OSP [4] - having different wafer yield and time complexities - have been proposed, but the current methodology of using a single algorithm for all chips on the production line is not sufficient for maximum yield or minimum repair time.

Since chips with similar failures are more likely to be repaired by similar algorithms, we propose a wafer-scale clustering module for classifying the chips on the basis of errors and wafer position, and a neural network based RA algorithm recommender module which would allow RA algorithm selection. To the best of the authors' knowledge, the proposed methodology to identify chips with similar failures in the Backend of the Line (BEOL) and improve the repair efficiency is the first of its kind. The longer the wafer is kept in the BEOL, more are the chances of its contamination. Thus decreasing the production time will allow manufacturing of more wafers and decreasing the probability of contamination.

The summary of our contributions is as follows.

- A wafer-scale clustering module for classifying the chips on the basis of position on the wafer and number of errors. This would allow classification of chips exhibiting specific faults into a single cluster.
- A neural network based Redundancy Analysis recommender module which would allow selection of an algorithm out of a suite of algorithms to analyze and improve the repair efficiency while decreasing the overall time taken to repair this memory.

Different solutions can be provided in different conditions, given the solution exists. Quality of solution and time taken by each RA algorithm also depends on type of chip failure patterns. The proposed model takes these factors into account along with chip errors and provide the best algorithm recommendation. It also predicts if the chip should be discarded thus saving valuable time on the manufacturing line which can be utilized to repair other chips.

The outline of the paper is as follows. In Section 2, we have discussed the existing RA algorithms along with the genetic algorithm. Section 3 describes the proposed model in detail. Finally, we have discussed the experimental results and conclusion in Sections 4 and 5 respectively.

978-1-6654-4175-9/21/$31.00 © 2021 IEEE

## II. Related Work

### A. Existing RA Algorithms

There have been various contributions in the field of RA algorithms. These algorithms can be classified into two categories, exponential and heuristic algorithms. Exponential algorithms will repair all the repairable chips, but in general, are too slow to be used on the production lines. Heuristic algorithms, on the other hand, trade-off yield with a faster execution time. Exponential algorithms include Fault-driven [5] which builds a tree with all the possible repairs. Branch-and-Bound [1] , PAGEB [1] and Faulty Line Covering algorithm [3] optimize the naive exponential algorithm, bringing down the overall time required to repair the chip without sacrificing the yield. FAST [6] groups the faults to further bring down the time complexity but it starts to sacrifice the yield.

Repair-Most [2] and Broadside Algorithm [2] try to repair the chip greedily. LECA [3] uses Effective Coefficients to rank the rows and columns of a chip in the order in which they have to be repaired. The One Side Pivot (OSP) [4] uses Pivot fault properties to find repair priorities reducing the analysis time even when the error rate is high. RA algorithm for chip repair must be chosen carefully as they provide varying yield and time complexity. Genetic Algorithm (GA) [7] is an evolutionary algorithm used to solve optimization problems. In [8], GA has been proposed as a possible solution to RA.

An ideal RA should find a repair solution whenever one exists. It should complete execution in a reasonable length of time and abort at the earliest sign of un-reparability.

### B. Learning Algorithms

**Neural Networks** (NN) [9] are a set of learning algorithms which are capable of processing dynamic state of external inputs. Neural networks learn or train by using input data features to extract meaning, relations and patterns. Typically NN are arranged in layers and each layer passes extracted information to the next layer for further computation. The processing is done using weighted connections and based on learning rule these weights are modified. NN follows universal approximation theorem i.e. it can approximate arbitrarily well for any complex relation between input and output.

**Decision trees** [10] are supervised machine learning technique which uses data feature to create decision rules. A tree consists of sequence of binary decisions which are created based on best single binary split. Best single binary split is determined by observing all splits and using the one with least error. Splitting can be stopped when there is single value feature in leaf node or can be controlled by user.

**Clustering** [11] is an unsupervised machine learning technique which can be used to group similar data within some threshold value. These similar data might also share other similar characteristics. Once clustering is performed the cluster parameters are used to interpret the data to gain additional insights.

## III. Proposed Model

### A. Overview

The proposed model illustrated in Figure 2 (A) consists of the wafer scale-clustering module, the Neural Network Recommender (NNR) module, and the Redundancy Analysis (RA) Simulator. The wafer scale-clustering module extracts features of the chips at the wafer level. The chip parameters are then augmented to the extracted features from the wafer scale-clustering module. The input parameters of the NNR module comprise of these combined parameters. This module predicts whether the chip is repairable or not. If the chip is repairable, an RA Algorithm is recommended which is used by the RA Simulator to perform chip repair simulation.

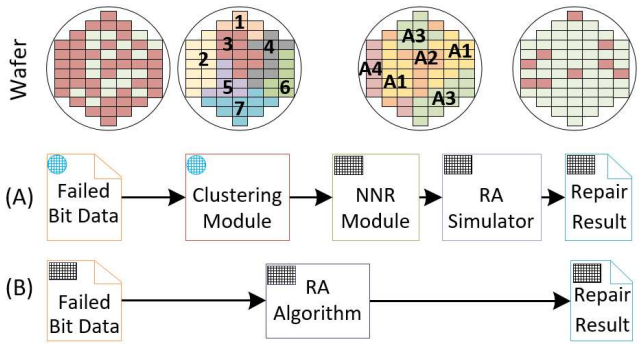


Fig. 2: Proposed and Existing Redundancy Analysis Model

Figure 2 (B) describes the traditional setup where failed bit data of individual chips are used as an input for the RA algorithm. In a traditional setup all processes are at a chip level, represented by the black chip in the diagram. Whereas in the proposed method, data extraction and clustering are done at a wafer level, illustrated by the blue wafer at the top left corners of the modules. Rest of the modules work at a chip level. The algorithm used by the setup is pre-selected with the individual chip repair result as output. The proposed model has an added advantage of analyzing the failed bit data for the whole wafer. This model selects the best RA algorithm from a class of predetermined algorithms.

### B. Wafer Scale Chip Clustering

Algorithm recommendation would require complete analysis and classification of chips but doing so individually would slow down the memory repair process. Hence, we introduce clustering which would allow classification of chips taking in very few parameters, with a low time overhead. Parameters like cluster size, position, and error count were used for classifying the chips. The following scoring formula is used for clustering based on distance and error:

$$S_i = \frac{E_i \times E_w}{\sum_{i=1}^{n} E_i} + \frac{D_i \times D_w}{\sum_{i=1}^{n} D_i} \quad (1)$$

The chip score $S_i$ determines which cluster the chip belongs to. Chips whose scores are closer to each other belong together in the same cluster. The error difference $E_i$ is the difference between chip error and mean error of the cluster. The Euclidian distance $D_i$ is the distance of the chip from the cluster centroid. The error and the distance weights $E_w$ and $D_w$ are weights of error and distance factors.

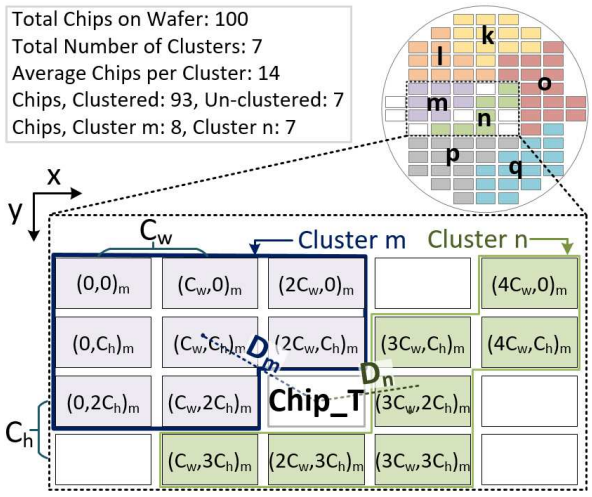


Fig. 3: An example of the clustering performed on a wafer

A common case example illustrated in Figure 3 represents a part of the wafer. The objective of this example is to classify *Chip_T*. Let the distance between centers of adjacent chips be $C_w$ (chip width) and $C_h$ (chip height). We define sets $X_j$ and $Y_j$, which consist of x and y coordinates of all the chips in cluster *j*. The Euclidean cluster centers can be calculated as:

$$(C_x, C_y) = \left(\frac{\sum X_j(i)}{|X_j|}, \frac{\sum Y_j(i)}{|Y_j|}\right) \quad (2)$$

Using, the mean (X, Y) coordinates of the chips in clusters *m* and *n*, and using (2), the cluster centers for *m* and *n* are:

$$C_x(m) = \frac{(3\times 0 + 3\times 1 + 2\times 2)C_w}{8} = \frac{7C_w}{8}$$
$$C_y(m) = \frac{(3\times 0 + 3\times 1 + 2\times 2)C_h}{8} = \frac{7C_h}{8}$$
$$C_x(n) = \frac{(1\times 1 + 1\times 2 + 3\times 3 + 2\times 4)C_w}{7} = \frac{20C_w}{7}$$
$$C_y(n) = \frac{(1\times 0 + 2\times 1 + 1\times 2 + 3\times 3)C_h}{7} = \frac{13C_h}{7}$$

Thus, based on the Euclidian distance, $D_m$ and $D_n$ of clusters m and n from Chip_T, this chip should be grouped in Cluster *n* as $D_m \cong 1.8 \cdot D_n$. If the chips in Cluster n belong to an arc type-specific error [12], if Chip_T error is not an arc error, this chip would be misclassified. This misclassification is used to calculate the weights in (1) to improve the clustering accuracy. The errors have to be considered as a factor when classifying the chip since mean error allows differentiation of chips in the specific error class from other chips.

Since Error and Distance are different units, we added a normalizing factor of summation of errors and distance in (1). We also introduce distance and error weights for accurately classifying chips based on both distance and error. To obtain an accurate value of these weights we use misclassified chips, based on clusters derived from specific errors in [12]. The same example illustrated in Figure 3 is used to calculate error weights. Since *Chip_T* has to be classified in cluster m, score $S_x(m)$ should be less than $S_x(n)$ giving us the following equation:

$$S_x(n) - S_x(m) > 0$$
$$\Rightarrow \frac{(avg(E_n) - avg(E_m)) \times E_w}{\sum_{i=1}^{n} E_i} + \frac{(D_n - D_m) \times D_w}{\sum_{i=1}^{n} D_i} > 0$$

In the example, $D_m \cong 1.8 \cdot D_n$. Generalizing this, let the correct cluster distance, $D_m \cong d_{Diff} \cdot D_n$, where $D_m$ and $D_n$ are the distances from the correct and misclassified clusters respectively. Thus, $D_n - D_m = (1 - d_{Diff}) \cdot D_n$. Similarly, the average error difference between the clusters equals $e_{Diff} \cdot E_{average}$, where $E_{average} = \sum E_i/n$ and $D_{average} = \sum D_i/n$.

$$\Rightarrow \frac{e_{Diff} \times E_{average} \times E_w}{\sum_{i=1}^{n} E_i} + \frac{(1 - d_{Diff}) \times D_n \times D_w}{\sum_{i=1}^{n} D_i} > 0$$
$$\Rightarrow \frac{e_{Diff} \times E_{average} \times E_w}{n \times E_{average}} > \frac{(d_{Diff} - 1) \times D_n \times D_w}{n \times D_{average}}$$
$$\Rightarrow E_w > (d_{Diff} - 1) \times D_w \times D_n / (D_{average} \times e_{Diff})$$

The calculation for the above example is repeated for multiple misclassified chips, thereby obtaining a set of suitable values for $E_w$ and $D_w$ to be used in the scoring equation maximizing the correct classifications. The K cluster medians are initialized inside the wafer at equal distances from each other in a grid. We experimentally found this method of cluster center assignment to be better than random or error value based assignment. The centers are considered stable if the number of changes of chip cluster in the last iteration is within a user-defined threshold.

Since we use a modified K-means clustering algorithm, the quality of clusters and clustering algorithms needs to be assessed. Purity [13] is the evaluation criterion used for finding clustering accuracy:

$$Purity = \frac{1}{N}\sum_{j=1}^{k} max_j(n_j^i) \quad (3)$$

Here N is the number of chips, *k* is the number of clusters, $n_j^i$ is the number of chips in cluster *j* with label *i*. The clustering algorithm provides size, position and mean error of the cluster as inputs to the neural network.

### C. Neural Network Recommender System

#### 1) Problem definition and setting

Different RA algorithms are suitable for different chip configurations and error patterns. Let, **A** represent the class of RA algorithms under consideration. $\mathbf{a_i} \in \mathbf{A}$ denotes the $i^{th}$ algorithm. Given a DRAM chip with an unseen failure configuration, we have to select an algorithm from **A** which will repair the chip using a combination of algorithm runtime, spare rows, and columns allocated.

#### 2) Parameters

We select the parameters based on the output of the clustering algorithm and the chip configuration such as the dimensions of the chip: number of rows, columns, number of spare rows and columns available and the number of faults in the chip. Based on the fault distribution, we then derive parameters by binning the number of faults in each row and column. We similarly bin repair coefficient values, calculated using the following equations:

$$f(i,j) = \begin{cases} 1, & if\ row\ i, column\ j\ is\ faulty \\ 0, & otherwise \end{cases}$$
$$RC_i = \sum_{j=1}^{columns} f(i,j) - \sum_{j=1}^{columns}\left(f(i,j) \times \sum_{i=1}^{rows} f(i,j)\right)$$

where $RC_i$ is the repair coefficient for *row i*. Similarly, $RC_j$ is calculated for the repair coefficients of each column.

*3) Data generation*

We derived training data from [12] using parameters from experimental data and simulated the algorithms in **A** on ~$4.3\times10^7$ DRAM chips, which resulted in the following input parameters for the recommender system:

S: Whether the algorithm was able to repair the chip
T: Time taken to repair the chip
$R_u$: Number of spare rows used by the algorithm
$C_u$: Number of spare columns used by the algorithm

If an algorithm does not repair the chip, it is assigned a score of -1, otherwise, the score is calculated using the simulation results coupled with hyperparameters.

$$a_{i_{score}} = \begin{cases} \varphi \times T - \omega \times R_u - \rho \times C_u, chip\ repaired \\ -1, \quad chip\ not\ repaired \end{cases}$$
$$a_t = \{a_i \mid a_i \in \mathbf{A}\ and\ \arg\max_i(a_{i_{score}})\} \quad (4)$$

where $\varphi$, $\omega$, $\rho$ are positive constants and $a_i \in \mathbf{A}$.

*4) Model*

We implemented the neural network model using PyTorch v1.0.1 [14]. The network consisted of eight hidden layers with 64, 128, 256, 512, 256, 128, 64 and 32 units with ReLU activation and a softmax output layer. We use a model dropout of 0.2 for the hidden layers. Five output units, 4 for the algorithms under consideration (Broadside [2], LECA [8], OSP [4] and Genetic [8]) and 1 node for the unrepairable chip class were used. If a chip is predicted as unrepairable, we directly discard that chip, thereby saving valuable time, which can be utilized to repair other chips. Experiments with larger networks, with more than eight hidden layers and more units in each layer, did not show any substantial advantage in the results. The objective of the recommender is to select the best RA algorithm for a particular chip.

## D. Other Methods

Decision tree will try to minimize entropy to get an optimum binary split. Since a decision tree is not very powerful on its own, random forest technique [10] is used. It uses ensemble learning i.e. it combines a number of decision trees to get a final decision. Random forest with 10 trees are used to develop an algorithm recommendation with entropy as criterion to measure the split quality.

Clustering allows a wafer level classification of chips, which can't be done through traditional chip level methods. To gauge the impact of clustering, we removed the features which contained specific details obtained from clustering. We ran the same neural network and fine-tuned the hyper parameters to get better results. A comparison of these methods has been done in the next section.

# IV. Experimental Results

## A. Wafer Scale Clustering

The result of chip simulation and clustering can be seen in Figure 4. Chip colors in Figure 4 (A) are based on relative error values based on simulated errors. In Figure 4 (B) the centroid errors of clusters are used to determine the range of chip colors. For clustering accuracy the following error labels were used:

- Specific error: Line, Arc or Segment [12]
- General error: High (>0.1% fault) and Low (≤0.1% fault)

In the illustrated test wafer, 3261 out of 3496 chips were classified correctly. Using (3) we obtained a very high purity of 0.933. Thus, the proposed mechanism allowed quick accurate clustering based on the 5 labels. Average silhouette scores of -0.305 and 0.414 were obtained using the naive K-means and the proposed methodology respectively. Thus, the proposed scoring mechanism allows better cluster formations for the purposes of wafer level pattern detection then naive K-means.

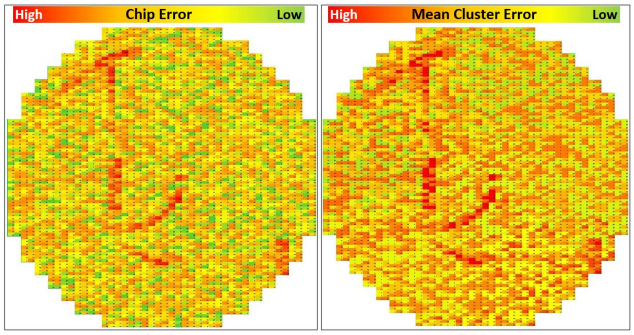


Fig. 4: (A) Chip Error in a cluster and (B) Mean Cluster Error

## B. Other Methodologies

We have performed a comparison of the three methodologies discussed in sub-section III-D. The proposed method was compared with a decision tree based solution and a neural network recommender system without clustering. In Figure 5, we observe that the decision tree converged to a maximum accuracy of 69.38%. The recommender without clustering had a high training accuracy, but the test accuracy of the model was only 33.66%. Our proposed solution converged to a testing accuracy of 76.0%, surpassing all the other tried methods.

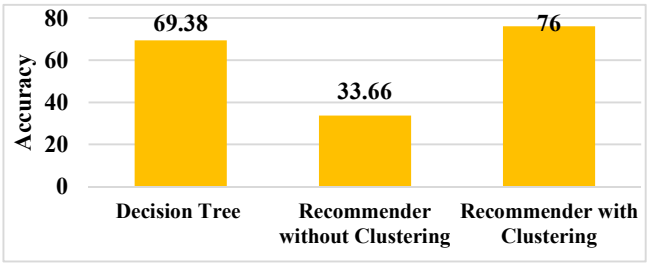


Fig. 5: Comparison between proposed and other methods (III-D)

## C. Neural Network based Recommender System

The yield of GA for the dataset was 98.99% and that of the recommender system was 91.12%. Nevertheless, in runtime comparison, the recommender system was 2.2x faster than the GA as illustrated in Figure 6. Time is of the essence while manufacturing wafers as more time on the manufacturing line, means higher chances of contamination.

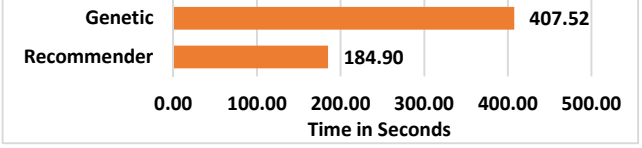


Fig. 6: Runtime comparison (seconds) between the Genetic Algorithm and the Recommender

The train-validation-test split was 80:10:10. The following hyperparameters were used in both the recommender with and without clustering: A batch size of 512 was selected to obtain a correct balance between the training time and accuracy. The model converged after 175 epochs. The gradient clipping and weight decay values were set to 0.2 and 0.00001 respectively. An Adam Optimizer was used due to low error rates. One cycle policy with maximum learning rate of 0.001 was used.

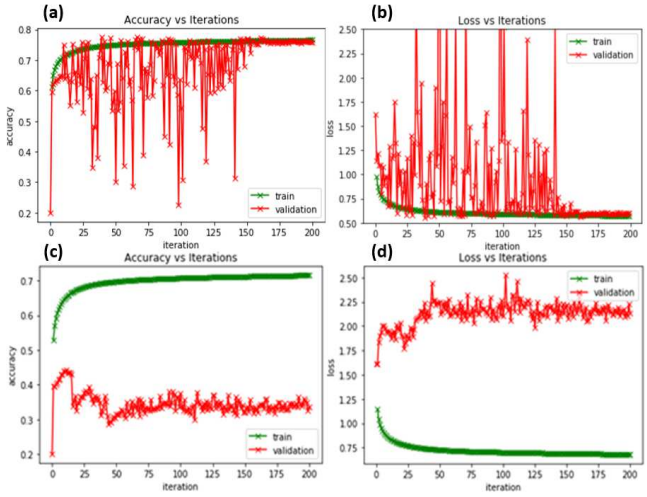


Fig. 7: Accuracy and Loss plots with and without clustering

We have illustrated the training iteration against accuracy and loss of the Neural Network with and without the clustering in Figure 7 (a), (b) and Figure 7 (c), (d) respectively. We observed after 200 epochs the model validation accuracy for the proposed recommender system converged to 75.92%. On the test set accuracy converged to 76.0%. The categorical cross-entropy loss function reduced the loss from 1.6224 to 0.5967. When we ran the optimized recommender without clustering, training the neural network for 200 epochs, it only converged to a validation accuracy of 33.63%. On the test set it was only able to converge to an accuracy of 33.66%. From Figure 7 (c) and (d), we observe a huge difference in the validation and training set accuracy as well as the loss when clustering was not used.

Table I, is the confusion matrix obtained for the 5 different categories by the proposed solution. We observe that the OSP algorithm has the lowest precision and recall as both OSP and Broadside are able to repair chips with a small number of errors. Thus, a large number of chips that can be repaired by the OSP are predicted as Broadside and vice-versa. Unsolvable chips are also predicted with a high precision as number of errors is a very strong indicator of a chip failure. For the Broadside, LECA and Genetic algorithm we observe a high rate of both precision and recall.

TABLE I. CONFUSION MATRIX FOR THE PROPOSED MODEL

| | Broadside | LECA | OSP | Genetic | Unsolvable | **Recall** |
|---|---|---|---|---|---|---|
| Broadside | 36716 | 1827 | 7005 | 952 | 1 | 78.96% |
| LECA | 374 | 39209 | 1952 | 3656 | 1052 | 84.79% |
| OSP | 12179 | 2313 | 28079 | 2623 | 1181 | 60.55% |
| Genetic | 92 | 7626 | 2059 | 35433 | 854 | 76.92% |
| Unsolvable | 0 | 1176 | 4718 | 2223 | 38152 | 82.46% |
| **Precision** | 74.38% | 75.18% | 64.09% | 78.94% | 92.51% | |

We have illustrated the recommender system results in Figure 8 by dividing the chip error rate into 3 sections, low, medium and high for error rates of less than 0.6%, between 0.6% and 1.2% and greater than 1.2% respectively. The selections by the recommender system can be seen in the Pie Charts. Broadside was the most popular choice at low error rates followed by LECA and OSP, and GA was not recommended for this error rate range. GA was preferred at high error rates, followed by OSP and LECA.

At low error rates, all the algorithms and the proposed model had 100% yield as the chips are easy to repair and often contain less number of row and column errors than the available spares. At medium error rates, the model outperformed existing heuristic algorithms in terms of yield by at least 9.1% and up to 36.8%. Clustering was able to identify specific faults at high error rates which allowed the model to outperform Broadside, LECA and OSP algorithms in terms of yield by at least 32.9%. Even though GA has a slightly higher wafer yield, the proposed model is 2.2x faster than the GA as illustrated in Figure 6.

With the help of the recommender system we were able to efficiently utilize the solving capabilities of different algorithms and repair a large number of chips with a similar average runtime as that of the heuristic algorithms.

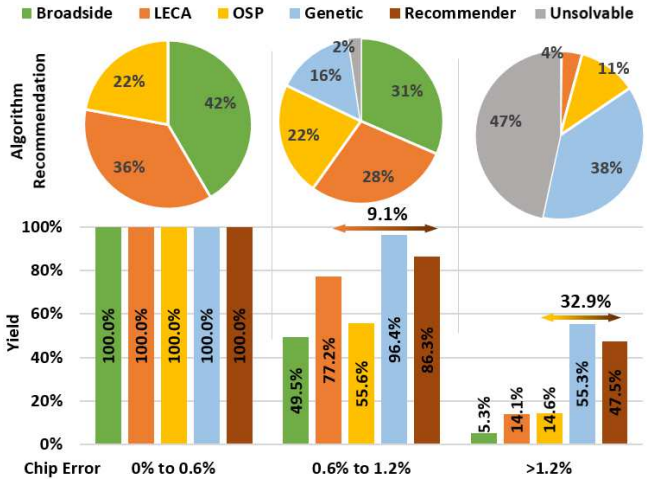


Fig. 8: Algorithm recommended by the proposed model and Heuristic algorithm vs recommender yield for different error ranges

## V. CONCLUSION

In this paper, we have proposed a model for improving the wafer yield while decreasing the wafer repair runtime. This solution can be used in the manufacturing line directly to improve the wafer yield. The model accuracy can be improved as more wafer data is available when more wafers are manufactured. We can also add more algorithms to the model with the advancement of new RA algorithms in this field. With the increasing density of DRAM devices, where error probabilities are even higher, heuristic algorithms will consume more time to repair the faulty memory. The proposed model will save time on the manufacturing line, thus allowing production of more wafers. Upon deployment the proposed model would contribute to the field of memory repair and redundancy analysis.

## REFERENCES


[1] S. K. Cho, K. Wooheon, C. Hyungjun, L. Changwook and K. Sungho, A Survey of Repair Analysis Algorithms for Memories, ACM Computer Survey, 2016.

[2] M. Tarr, D. Boudreau and R. Murphy, "Defect analysis system speeds test and repair of redundant memories," *Electronics,* vol. 29, no. 12, pp. 175-179, Jan. 1984.

[3] F. Lombardi and W. K. Huang, "Approaches for the repair of VLSI/WSI RRAMs by row/column deletion," *International Symposium on Fault-Tolerant Computing.,* pp. 342-347, 1988.

[4] J. Kim, K. Cho, W. Lee and S. Kang, "A new redundancy analysis algorithm using one side pivot," *International SoC Design Conference (ISOCC),* pp. 134-135, Jeju, 2014.

[5] J. R. Day, "A fault-driven comprehensive redundancy algorithm," *IEEE Des. Test Comput.,* vol. 2, no. 3, p. 35–44, Jun. 1985.

[6] C. Hyungjun, W. Kang and S. Kang, "A fast redundancy analysis algorithm in ATE for repairing faulty memories," *ETRI J.,* vol. 34, no. 3, pp. 478-481, June 2012.

[7] K.-F. Man, K.-S. W. Tang and T. W. S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Transactions on Indus-trial Electronics,* vol. 43, no. 5, pp. 519-534, Oct. 1996.

[8] J. Milbourn, "Strategies for Optimising DRAM Repair," *Durham theses, Durham University,* 2010.

[9] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning," *Nature,* vol. 521, p. 436–444, 2015.

[10] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, p. 5–32, 2001.

[11] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability,* vol. 1, p. 281–297, 1967.

[12] Atishay, A. Gupta, R. Sonawat, H. K. Thacker and B. Prasanth, "SEARS: A Statistical Error and Redundancy Analysis Simulator," *27th International Conference on VLSI-SoC,* pp. 117-122, 2019.

[13] J. Wu, H. Xiong, J. Chen and W. Zhou, "A Generalization of Proximity Functions for K-Means," *Seventh IEEE International Conference on Data Mining,* pp. 361-370, 2007.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan and et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library.," *Advances in Neural Information Processing Systems 32, Curran Associates, Inc.,* p. 8024–35, 2019.