QUANTILE ONLINE LEARNING FOR SEMICONDUCTOR FAILURE ANALYSIS

Bangjian Zhou[⋆]*, Pan Jieming*[†]*, Maheswari Sivan*[†]*, Aaron Voon-Yew Thean*[†]*, J. Senthilnath*[⋆]

[⋆]Institute for Infocomm Research (I^{2}R), A*STAR, Singapore
[†]Electrical and Computer Engineering, National University of Singapore, Singapore

ABSTRACT

With high device integration density and evolving sophisticated device structures in semiconductor chips, detecting defects becomes elusive and complex. Conventionally, machine learning (ML)-guided failure analysis is performed with offline batch mode training. However, the occurrence of new types of failures or changes in the data distribution demands retraining the model. During the manufacturing process, detecting defects in a single-pass online fashion is more challenging and favoured. This paper focuses on novel quantile online learning for semiconductor failure analysis. The proposed method is applied to semiconductor device-level defects: FinFET bridge defect, GAA-FET bridge defect, GAA-FET dislocation defect, and a public database: SECOM. From the obtained results, we observed that the proposed method is able to perform better than the existing methods. Our proposed method achieved an overall accuracy of 86.66% and compared with the second-best existing method it improves 15.50% on the GAA-FET dislocation defect dataset.

***Index Terms*—** Online learning, single-pass, deep neural network, semiconductor failure analysis

1. INTRODUCTION

For several decades the semiconductor industry follows Moore's law to drive development, and billions of well-designed semiconductor chips are put into use every year. Fine chips require more care about defects. Notably, as the devices are subjected to aggressive channel length scaling, the impact of bridge and dislocation defects on the transport properties of the transistor becomes aggravated. As a result, engineering efforts are essential for semiconductor failure analysis (FA) to detect defects on time and guarantee major components work rightly [1].

Lately, Machine Learning (ML) has become increasingly important to accelerate semiconductor FA. There are mainly 2 kinds of the semiconductor dataset used in the ML model to learn and generalize, (1) Unstructured dataset, for example, wafer map imagery used to understand the wafer quality and identify the defects patterns [2] and (2) Structured dataset, for example, the transfer characteristics (drain current (I_d) vs. gate voltage (V_g)) of the transistor for different position of bridge defects [3]. The structured dataset is favoured by researchers as the features like slope and intercept (for example, sub-threshold swing and threshold voltage for transistor) represent healthy/unhealthy states more discriminatively. The structured datasets' features differ a lot in ranges, which is attributed to the operational principle of a transistor as a logic switch where the drain current varies across $\sim$8 orders of magnitude, whereas drain voltage varies from 0-1V (for N-channel Metal-oxide Semiconductor).

The ML models used for the FA can be divided as traditional ML, such as tree-based model [3], Neuro-fuzzy system (NFS) [4], and Deep Neural Network (DNN) model [2]. The DNN models are affected by the high-range features that would take a dominant place, making other features insignificant. Hence, data normalization is necessary for DNN training. Meanwhile, as the number of semiconductor chips is increasing exponentially, the scenario has changed. The offline batch-mode DNN training fails to keep up pace with the large data stream, online single-pass mode DNN training is preferred. This makes the data normalization for the DNN difficult. Though we have many well-designed methods like min-max scaling, z-score normalization [5] and quantile normalization [6] for offline training, whereas in the online scenario, we lack global statistics requested by the normalization methods. Although some researchers have proposed using sliding windows to get statistics or designing equations/layers to approximate the statistics used in the normalization, most online normalization methods are still designed in batch-mode, whereas single-pass mode is rarely adopted.

In this paper, we propose a novel quantile online learning method for semiconductor FA. The main contributions of this study are summarized as follows: (1) Unlike quantile normalization averaging the feature values in batch/mini-batch setting with the same quantile value assigned for different samples, we use quantile numbers to replace raw features. Here, we observe the proposed approach is suitable for semiconductor datasets with varying feature ranges. (2) We trained the online DNNs in a purely single-pass mode including the input data normalization, there are very few methods

This study is supported by the Accelerated Materials Development for Manufacturing Program at A*STAR via the AME Programmatic Fund by the Agency for Science, Technology and Research under Grant No. A1898b0043.

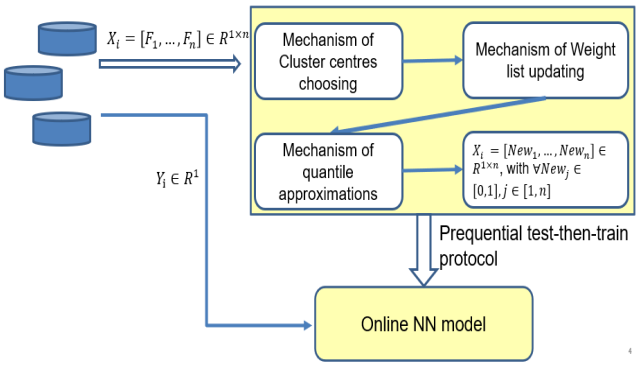

Fig. 1. The work flow of the quantile online learning method

performing semiconductor FA in such a way. (3) We propose a discount factor to control the proportion of local or global information in the statistics, which helps with defect detection. (4) Our experimental results demonstrate that the proposed method perform better than the existing methods on four semiconductor defect detection datasets, which exceeds the second-best existing method by 15.50%.

2. RELATED WORK

ML-guided FA. Many traditional ML-based approaches have been applied to semiconductor FA. He et al., [7] utilized a k-nearest neighbor (KNN) rule-based algorithm to handle faults. Xie et al., [8] applied a support vector machine to detect defect patterns in the image. Nawaz et al., [9] developed a Bayesian network for etching equipment fault detection. After Lecun et al., [10] pointed out that DNN can improve the state-of-the-art methods in many fields, researchers also tried DNN methods, while the traditional methods are still common. Hsu et al., [11], Kim et al., [12] and Chien et al., [2] proposed some variants of convolutional neural network for FA in image datasets. Pan et al., proposed a tree-based transfer learning model from FinFET to GAA-FET [3]. Kim et al., [13] devised a generative model to handle process drift in the defect detection scenario. Ferdaus et al., [4] devised an online NFS for semiconductor FA. However, most methods are focused on offline settings, this motivates us to design an efficient online learning methods for DNNs.

Input data normalization. Commonly used offline input data normalization strategies are min-max scaling, z-score normalization and decimal scaling [5]. However, these techniques need to adapt to online scenarios and adaptive versions are few. To the best of our knowledge, the adaptive normalization methods can be divided into two categories: (1) Ogasawara et al., [14], Gupta et al., [15] used sliding window to adapt statistics in streaming data. (2) Passalis et al., [16] proposed DAIN that uses additional linear layers to mimic the adaptive mean and standard deviation value for normalization. Tran et al., [17] and Shabani et al., [18] proposed improved variants of DAIN to handle time-series data. Though these methods were devised for online settings, they are designed in batch-mode. Our proposed method is a single-pass design, which means there is hardly any latency in getting predictions. And quantile normalization is another extremely popular method that could produce well-aligned distributions so all samples are from the same distribution, which is commonly seen in gene expression dataset [19], [6]. Note, our proposed method is totally different from quantile normalization; as they would not use the quantile number to replace the feature value.

3. QUANTILE ONLINE LEARNING

The proposed quantile online learning (QOL) classifies semiconductor FA datasets on the fly. The flow diagram of the proposed QOL is shown in Fig. 1 and discussed in detail in the below subsections.

3.1. Problem statement

The semiconductor structured input data stream i.e, $X_i = [F_1, ..., F_n] \in \Re^{1\times n}$, $i \in \mathcal{N}$, which means i^{th} sample and n is the number of dimensions. Each input has a corresponding label $Y_i \in \mathcal{N}$ that represents the defect types; our objective is to address the classification problem of the defect types. To mimic real-world settings, a prequential test-then-train protocol is applied. This protocol would use the model to give predictions and then do the training.

3.2. Buffer list initialization

Initially, a small list of samples is collected i.e, $C = [X_1, ..., X_p] \in \Re^{p\times n}$, where p is the number of samples, set between 10 to 30. We collect the feature values in a 2D buffer list from samples C i.e, V = $[[V_{11}, .., V_{1p}], ..[V_{n1}, .., V_{np}]]$. The V_{ij} means the feature value F_i from sample X_j. Next, we sort each single 1D list independently and define another 2D weight list $W = [[1, .., 1], .., [1, .., 1]]$ with all ones. We treat the buffer list $V = \{V_{ij}\}^{n\times p}$ as cluster centres which capture the knowledge of the distribution of raw data, and weight list $W = \{W_{ij}\}^{n\times p}$ as the number of samples in the cluster. During this phase, we wait for p samples to start, whereas the remaining steps use single-pass mode.

3.3. Cluster centre choosing

We obtain the initial V and W and the raw input features X_i from the data stream. We choose the cluster centres for each feature by calculating distance $dis(V_{ij}, F_i) = ||V_{ij} - F_i||_2 = \sqrt{(V_{ij} - F_i)^2}$ and find the index of minimum distance for the k^{th} feature using

$$idx_k = \{i | \min dis(V_{ki}, F_k), 1 \leq i \leq p\},$$

The index list $idx = [idx_1, ..., idx_n]$ is used as a reference to update later steps.

3.4. Weight list updating

The main idea behind updating is to replace the old cluster centre with the new feature value and make the corresponding weight contribute to the closest cluster centres. We consider 2 cases in updating: (i) $F_i \geq V_{i,idx_i}$ and (ii) $F_i < V_{i,idx_i}$. For case (i) in Fig 2(a), $old\_weight = W_{i,idx_i}$ represents the weight for V_{i,idx_i} before updating. We first calculate a $percentage$ factor as below:

$$percentage = \frac{dis(V_{i,idx_i}, F_i)}{dis(V_{i,idx_i-1}, F_i)} \tag{1}$$

Then we update $W_{i,idx_i}, W_{i,idx_i-1}$ using,

$$W_{i,idx_i} = (1 - percentage) \times old\_weight + 1 \tag{2}$$

$$W_{i,idx_i-1} = W_{i,idx_i-1} + percentage \times old\_weight \tag{3}$$

These equations make sure the closer the value is to the cluster centres, the more weightage is assigned from the old weight. If the W_{i,idx_i-1} does not exist, we give all weight to the new cluster using,

$$W_{i,idx_i} = old\_weight + 1 \tag{4}$$

For case (ii) in Fig.2(b), we still use old_weight as above, and only give modifications to the above equations as follows:

$$percentage = \frac{dis(V_{i,idx_i}, F_i)}{dis(V_{i,idx_i+1}, F_i)} \tag{5}$$

$$W_{i,idx_i} = (1 - percentage) \times old\_weight + 1 \tag{6}$$

$$W_{i,idx_i+1} = W_{i,idx_i+1} + percentage \times old\_weight \tag{7}$$

For the specific case W_{i,idx_i+1} does not exist, then we use Eq.(4) to update. Next, we introduce a discount factor η, which is used to control the forget rate. To make the buffer list capture a sense of local trend, we set η between 0 to 1. After each sample's all features modification, we would update the whole weight list $\{W_{ij}\}^{n\times p} = \eta \times \{W_{ij}\}^{n\times p}$. Finally, we assign $V_{i,idx_i} = F_i$ as the new cluster centre.

3.5. Quantile approximations

After we find the suitable position for the current sample's feature values, we replace all the feature values F_i using,

$$F_i = \sum_{j=1}^{idx_i} W_{ij} / \sum_{j=1}^{p} W_{ij} \tag{8}$$

The weight for each cluster centres is used to approximate the number of feature values that are less than it so that we obtain an approximation of the quantile for each feature value. Then we feed the normalized data $X_i = [New_1, ..., New_n]$ to train the online DNN model.

Algorithm 1: Quantile online learning

Input: sample from data stream $\mathbf{X_i}$ with label Y_i, buffer list length p, discount factor η

Output: Normalized data stream $\hat{\mathbf{X}}_\mathbf{i}$, trained online NN model $\mathbf{H}$, predictions $\hat{Y}_i$ of sample X_i

Initialize the buffer list V and weight list W (sec 3.2)

while X_i *is not None* **do**

  for $j = 1$ *to* n **do**

    calculate the minimum distance between F_j and $[V_{j1}, ..., V_{jp}]$, get index idx_j (sec 3.3)

    if $F_j \geq V_{j,idx_j}$ **then**

      Use Eqs.(1-4) to update $\{\mathbf{W_{jk}}\}^{\mathbf{1\times p}}$,$1 \leq k \leq p$

    if $F_j < V_{j,idx_j}$ **then**

      Use Eqs.(4-7) to update $\{\mathbf{W_{jk}}\}^{\mathbf{1\times p}}$,$1 \leq k \leq p$

    Calculate $newF_j$ with Eq.(8),

    $\hat{\mathbf{X}}_\mathbf{i}[\mathbf{j}] = \mathbf{newF_j}$

  $W = \eta \times \{W_{ij}\}^{n\times p}$

  Train the NN model with the test-then-train protocol using normalized data $\hat{\mathbf{X}}_\mathbf{i}$

Table 1. The detailed information about datasets

Dataset	dimension	classes	number of samples
GAA-FET-b	23	12	3051
GAA-FET-d	40	9	90
FinFET-b	23	10	2739
SECOM	590	2	1567

4. EXPERIMENTAL RESULTS

Data description. In this work, three real-world multi-class semiconductor fault detection datasets and a public dataset SECOM [20] from the UCI machine learning repository are considered. The real-world datasets include bridge defects in GAA-FET (GAA-FET-b), dislocation defects in GAA-FET (GAA-FET-d) and bridge defects in FinFET (FinFET-b). These FETs datasets were generated using a full three-dimensional technology computer aided design (TCAD) digital twin model and then were calibrated with the Current-Voltage curve in the actual device defects [3]. The SECOM dataset consists of complex signals for the engineers to analyze the defects. The detailed information of each dataset is shown in Table 1. Classifying imbalance classes in the dataset is more challenging as the minority class would only be $3 \sim 20\%$ of the majority class, say, SECOM is 104 fails to 1463 pass, GAA-FET-b is 27 to 764 and FinFET-b is 79 to 512. As a result, overall accuracy and class balanced accuracy were used as evaluation metrics.

Experiment setting. We chose the Adaptive Normalization (AN) from [15] and Deep Adaptive Input Normalization (DAIN) from [16] as baseline methods to compare the data

(a) case(i) idea (b) case(ii) idea

Fig. 2. The general ideas for weight list updating

normalization, and passing the raw data. As the baseline methods are variants of min-max and z-score normalization, which cannot handle feature values with outliers pretty well. Hence, we additionally pre-processed the data by logarithm transformation to make the feature range small, which is not adapted for our proposed method. We chose three online DNN models, which include a DNN with 2 layers fully connected feed-forward network, Deep Evolving Denoising Autoencoder (DEVDAN) from [21] and Autonomous Deep Learning (ADL) model from [22] to classify the normalized data in a single-pass mode. The ADL and DEVDAN are two evolving architectures designed for online settings, which would autonomously generate the needed hidden nodes. To make a fair comparison, we limited different models to all have 2 layers and the nearly same amount of parameters. We intended to prove our method's robustness by using these models. We applied the prequential test-then-train protocol to the whole datasets, except for GAA-FET-d as current online learning models rarely get good performance in limited sequences (90 samples). We split and augment the train set and evaluate the test set with the test-then-train protocol.

We fine-tuned the hyper-parameters in the following range: [0.1,0.5] as lr_{disc} and [$1e^{-3}$,$1e^{-1}$] as lr_{gen} for DEVDAN [21]; [0.05,0.35] as lr for ADL [22]; [$1e^{-4}$,$1e^{-1}$] as lr for DNN; batch size 2 to 5 and threshold δ 0.1 to 0.3 for AN [15]; $mean\_lr$ in [$1e^{-8}$,$1e^{-1}$], $scale\_lr$ in [$1e^{-8}$,$1e^{-1}$], $gate\_lr$ in [$1e^{-3}$,1] and batch size 5 for DAIN [16]. We select the batch size to be small so that it's a relatively fair comparison with our single-pass method.

Performance comparisons. We compare the performance of 3 data normalization approaches (AN, DAIN, QOL) and Raw data on semiconductor datasets that are combined with 3 DNN models separately. For the 3 multi-class semiconductor datasets, from Table 2, we observe that our QOL performs well for both overall and balanced accuracy while the AN was the second best, and even got better overall accuracy in GAA-FET-b. However, the DAIN method has low performance, due to (1) the data sequence being limited, and DAIN's converge rate being affected by more network parameters. (2) a small batch size may not be suitable for this method. Compared with the Raw data, AN and QOL's result is a strong demonstration of the need for data normalization. For the SECOM, we emphasize the balanced accuracy here as most methods can get high overall accuracy by predicting all as the majority. Our proposed QOL achieves the best balanced accuracy, while the DAIN, as well as AN, also performed better for this high-dimensional complex dataset. All the normalization methods can address the imbalance problems a bit, which may be due to normalized features being more discriminative. Among all the datasets, we find that our single-pass QOL is more stable in different settings and gets comparable or better performance than these batch-mode normalization methods.

Table 2. Classification accuracy in semiconductors FA

Dataset	pre-process	Overall accuracy/balanced accuracy		
		DNN	ADL	DEVDAN
GAA-FET-b	Raw	81.61/62.21	79.06/48.96	78.72/66.86
	AN	74.73/44.45	87.00/66.70	**86.95**/66.32
	DAIN	62.66/38.39	74.56/56.34	74.68/54.28
	QOL	79.15/53.29	85.61/**67.73**	85.80/69.26
GAA-FET-d	Raw	51.11/57.03	42.22/45.18	40.00/43.70
	AN	53.33/55.18	54.66/60.00	49.99/55.92
	DAIN	43.33/40.00	32.96/32.22	43.33/45.18
	QOL	**98.88/99.26**	**98.88/99.26**	97.77/98.52
Fin-FET-b	Raw	70.85/63.02	71.55/70.53	83.64/80.69
	AN	72.05/61.44	91.86/87.27	92.02/87.65
	DAIN	58.40/46.49	83.15/81.05	82.86/80.59
	QOL	76.01/68.63	**92.56**/88.74	92.83/**88.92**
SE-COM	Raw	92.53/51.34	**92.26**/51.11	92.47/51.22
	AN	90.72/52.87	90.81/52.92	92.29/51.66
	DAIN	91.90/51.18	90.55/53.32	90.55/52.60
	QOL	92.44/51.65	90.78/**53.53**	92.17/52.22

5. CONCLUSIONS

In this paper, we proposed a novel quantile online learning approach to address single-pass online classification for semiconductor FA. QOL utilizes the single-pass online framework where quantile representation is passed to DNN architecture to adapt to the change in the feature distribution of different semiconductor defects by prequential test-then-train protocol. This helps in achieving better classification performance in both overall accuracy and balanced accuracy. We compared the performance of our technique with the existing methods for a binary class and three multi-class semiconductor defect datasets, for all these datasets the proposed method performed better in comparison with the existing methods.

6. REFERENCES

[1] Y-J. Park, S-K. S. Fan, and C-Y. Hsu, "A review on fault detection and process diagnostics in industrial processes," *Processes*, vol. 8, no. 9, pp. 1123, 2020.

[2] J-C. Chien, M-T. Wu, and J-D. Lee, "Inspection and classification of semiconductor wafer surface defects using cnn deep learning networks," *Applied Sciences*, vol. 10, no. 15, pp. 5340, 2020.

[3] J. Pan, K.L. Low, J. Ghosh, J. Senthilnath, Md M Ferdaus, S.Y. Lim, E. Zamburg, Y. Li, B. Tang, X. Wang, J.F. Leong, S. Ramasamy, T. Buonassisi, C.-K. Tham, and A. V-Y. Thean, "Transfer learning-based artificial intelligence-integrated physical modeling to enable failure analysis for 3 nanometer and smaller silicon-based cmos transistors," *ACS Applied Nano Materials*, vol. 4, no. 7, pp. 6903–6915, 2021.

[4] Md M. Ferdaus, B. Zhou, J. W. Yoon, K.L. Low, J. Pan, J. Ghosh, M. Wu, X. Li, A. V-Y. Thean, and J. Senthilnath, "Significance of activation functions in developing an online classifier for semiconductor defect detection," *Knowledge-Based Systems*, vol. 248, pp. 108818, 2022.

[5] S. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *arXiv preprint arXiv:1503.06462*, 2015.

[6] Y. Zhao, L. Wong, and W. W. B. Goh, "How to do quantile normalization correctly for gene expression data analyses," *Scientific reports*, vol. 10, no. 1, pp. 1–11, 2020.

[7] Q. Peter He and J. Wang, "Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 20, no. 4, pp. 345–354, 2007.

[8] L. Xie, R. Huang, N. Gu, and Z. Cao, "A novel defect detection and identification method in optical inspection," *Neural Computing and Applications*, vol. 24, no. 7, pp. 1953–1962, 2014.

[9] J. M. Nawaz, M. Z. Arshad, and S. J. Hong, "Fault diagnosis in semiconductor etch equipment using bayesian networks," *Journal of Semiconductor Technology and Science*, vol. 14, no. 2, pp. 252–261, 2014.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] C-Y. Hsu and W-C. Liu, "Multiple time-series convolutional neural network for fault detection and diagnosis and empirical study in semiconductor manufacturing," *Journal of Intelligent Manufacturing*, vol. 32, no. 3, pp. 823–836, 2021.

[12] E. Kim, S. Cho, B. Lee, and M. Cho, "Fault detection and diagnosis using self-attentive convolutional neural networks for variable-length sensor data in semiconductor manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 3, pp. 302–309, 2019.

[13] Y. Kim, H. Lee, and C. O. Kim, "A variational autoencoder for a semiconductor fault detection model robust to process drift due to incomplete maintenance," *Journal of Intelligent Manufacturing*, pp. 1–12, 2021.

[14] E. Ogasawara, L. C. Martinez, D. De Oliveira, G. Zimbrão, G. L. Pappa, and M. Mattoso, "Adaptive normalization: A novel data normalization approach for non-stationary time series," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2010, pp. 1–8.

[15] V. Gupta and R. Hewett, "Adaptive normalization in streaming data," in *Proceedings of the 2019 3rd International Conference on Big Data Research*, 2019, pp. 12–17.

[16] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Deep adaptive input normalization for time series forecasting," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3760–3765, 2019.

[17] D. T. Tran, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Data normalization for bilinear structures in high-frequency financial time-series," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 7287–7292.

[18] M. Shabani, D. T. Tran, M. Magris, J. Kanniainen, and A. Iosifidis, "Multi-head temporal attention-augmented bilinear network for financial time series prediction," *arXiv preprint arXiv:2201.05459*, 2022.

[19] K. D. Hansen, R. A. Irizarry, and Z. Wu, "Removing technical variability in rna-seq data using conditional quantile normalization," *Biostatistics*, vol. 13, no. 2, pp. 204–216, 2012.

[20] M. Michael and J. Adrian, "UCI repository of SECOM Data set," *https://archive.ics.uci.edu/ml/datasets/SECOM*, 2008.

[21] A. Ashfahani, M. Pratama, E. Lughofer, and Y-S. Ong, "Devdan: Deep evolving denoising autoencoder," *Neurocomputing*, vol. 390, pp. 297–314, 2020.

[22] A. Ashfahani and M. Pratama, "Autonomous deep learning: Continual learning approach for dynamic environments," in *Proceedings of the 2019 SIAM international conference on data mining*. SIAM, 2019, pp. 666–674.