EasyChair Preprint

№ 15192

Wafer-View Defect-Pattern-Prominent GDBN Method Using MetaFormer Variant

Shu-Wen Li, Chia-Heng Yen, Shuo-Wen Chang, Ying-Hua Chu, Kai-Chiang Wu and Chia-Tso Chao

October 6, 2024

Wafer-View Defect-Pattern-Prominent GDBN Method Using MetaFormer Variant

Shu-Wen Li*, Chia-Heng Yen*, Shuo-Wen Chang‡, Ying-Hua Chu‡,
Kai-Chiang Wu* and Mango Chia-Tso Chao†
*Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
†Institute of Electronics, National Yang Ming Chiao Tung University, Hsinchu, Taiwan
‡Qualcomm Semiconductor Limited, Hsinchu, Taiwan

***Abstract*—Good-Die-in-Bad-Neighborhood (GDBN) is a technique employed to identify chips that pass initial tests but may have defects. Previous research used neural networks and expanded observation windows but ignored the impact of isolated dice. This paper improves wafer pattern information through denoising and creates a lightweight model. It also reduces training time by annotating multiple dice simultaneously. Experiments on real-world datasets show the model effectively captures more Test Escapes, reducing Defective Parts Per Million (DPPM) and improving return merchandise authorization gains.**

I. Introduction

Minimizing customer returns for reliability-driven IC products is crucial. While increasing fault coverage in tests is important, practical limitations mean 100% coverage is impossible. To address this, engineers identify and discard suspicious parts using methods like GDBN (good-die-in-bad-neighborhood) screening to prevent customer returns.

Instead of using a single predefined metric like bad neighbor count to represent a DUI's suspiciousness, some GDBN methods [1] [2] [3] [4] applied machine-learning (ML) techniques to learn a more complexed suspiciousness model with multiple input features based on silicon data set with actual pass/fail result as the label of each sample. Among those ML-based GDBN methods, [1] used bad neighbor counts in distance-based regions as input features and applied linear regression to model suspiciousness. [2] further used the bad neighbor count for each fail bin and each distance region to the DUI, the DUI's shortest distance to a wafer edge, and the probability of having a bad die at each location across different wafers as the input features while applying SVM regression for training.

Rather than using various bad neighbor counts as input features, [3] directly encoded each die location as a pixel representing its testing result, using the image formed by each pixel in the 7×7 observation window as the input features, and then trained the suspiciousness model with an MLP (multi-layer perceptron). Image-based encoding allows the model to learn graphical patterns of bad dice and determine the suspiciousness of a DUI by analyzing pass/fail distribution, which is more effective. With another image-based encoding, [4] further increased the observation window to the whole wafer, added the information of the wafer-level defect pattern into the input image encoding, and then utilized MobileNet [5] as the model for determining suspiciousness. A broader view of the complete wafer, along with an identified defect pattern, can help the model outperform [3].

In this paper, following the same input features as [4], we propose a defect-pattern-prominent GDBN method using Metaformer variant as its model. Our method contains the following three major differences to [4]. Based on [4], the wafer defect pattern is an effective feature for GDBN screening. To enhance this further, we use a denoising scheme to highlight major defect patterns on a wafer. This will improve the model's ability to identify common defect patterns. Second, Metaformer [6] has been proven more effective than MobileNet [5] on various vision tasks, and on top of this advanced model structure, we further develop a variant of it with direct-mapping block connection to better fit this GDBN screening problem. Thirdly, we develop a training scheme that can simultaneously label multiple DUIs for training such that the training efficiency and effectiveness can both be improved.

This paper utilizes an open dataset [7] released by TSMC in 2015. This dataset contains 172,950 labeled and 638,507 unlabeled wafer products. Experiments on this dataset showed that our GDBN method outperforms previous approaches, improving performance by 29.6%.

II. Proposed Methodology

A. Overview

Fig. 1 depicts the overall flow of our proposed methodology. With a labeled wafer map containing the test results of each die, and the defect-pattern type to which the wafer belongs, we apply a series of data preprocessing steps. Firstly, the data is padded into a 64×64 image. The second step is data denoising (detailed later in Section II-B), which removes bad dice that do not belong to any defective cluster. In the third step, we annotate multiple dice as DUIs. Any dice annotated as DUI will be considered good dice regardless of its testing result. This strategy enables our model (detailed later in Section II-C) to learn to detect test escapes from good dice in bad neighborhoods (detailed later in Section II-D). The final step is data encoding as presented in [4]. The encoded data shown in the upper right of Fig. 1 has eleven channels. The first three channels are based on die-level information, showing if the die is an out-of-wafer die, a good die, or a DUI. The remaining eight channels are derived from wafer-level information, representing eight different types of wafer defect

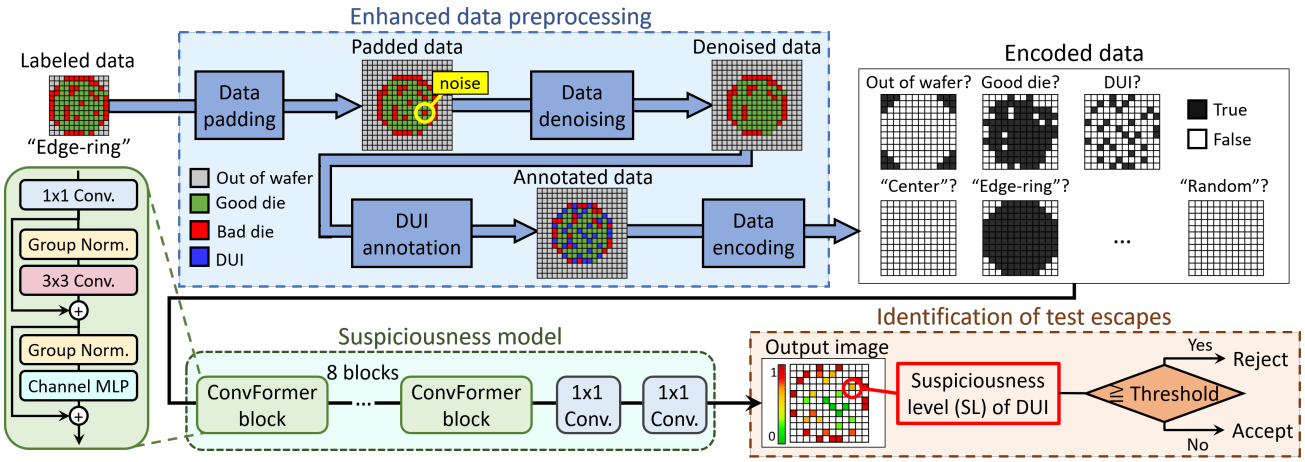

Fig. 1. The overall flow of the proposed methodology

types. Based on the defect pattern of the labeled data, we mark all of the dice inside the wafer in the corresponding channel. Only one out of these eight channels will be marked (assigned true); all of the other seven channels will be completely unmarked (assigned false).

Next, we split the encoded data into training and testing sets at an 8:2 ratio while maintaining the same proportion of data belonging to each type of wafer-level defect pattern, following the concept of stratified split. After training the model, we input the testing data to obtain the suspiciousness level (SL) of each DUI and set a threshold to make a judgment. If the suspiciousness level of the DUI is greater than or equal to the threshold, the DUI is considered a test escape and is removed; otherwise, it is retained. If the threshold is set lower, we can capture more test escapes and lower DPPM, indicating higher-quality products. However, this also increases the risk of mistakenly identifying dice as test escapes, leading to larger losses. This tradeoff will be discussed further in Section IV.

B. Denoise

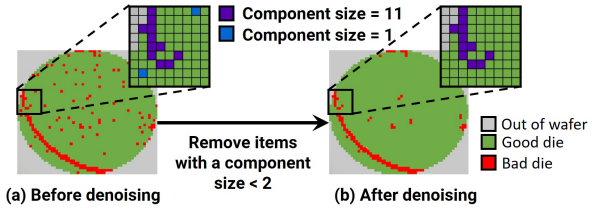

Fig. 2. Real example of before and after denoising

Based on the fact that defects tend to cluster together and form specific patterns, [4] has demonstrated that incorporating features with the whole wafer's test results and wafer defect pattern can effectively improve the capacity of GDBN. When analyzing defects on a wafer, it is crucial to understand that not all defects belong to a specific cluster. Isolated defects not only fail to contribute to the detection of latent defects, our primary objective, but also confound the model's attention from major clusters, thereby diminishing the method's overall effectiveness. To mitigate this issue, it is imperative to develop a methodology for identifying and eliminating these isolated defects (called noise).

To identify noise, We employ connected component labeling (CCL), a technique for detecting and labeling groups of connected pixels in binary images, where pixels are classified as either foreground (typically represented as 1) or background (typically represented as 0). The term "connectivity" refers to the relationship between adjacent foreground pixels. Pixels are considered connected if any of the eight neighboring pixels (including diagonals) within a 3×3 window is also a foreground pixel.

Typically, clusters of connected pixels indicate the same group. CCL assigns a distinctive label to each connected component in the binary image, ensuring that pixels with identical labeling belong to the same connected component. In our case, bad dice are marked as foreground, and the other are marked as background. After labeling dice, we calculate the number of dice in each component, referred to as the component size. If the component size is smaller than a certain threshold, we consider it as noise and remove it. In our proposed preprocessing method, we use 2 as the threshold for detecting noise.

Fig. 2 displays two wafer maps side by side. Green on the wafer map represents good dice within the wafer, while red represents bad dice. The gray area represents dice that are out of the wafer. The left figure shows the wafer map before denoising, while the right one shows the wafer map after denoising. Bad dice with a component size less than 2 in the left wafer map are considered noise and have been removed during processing. This ensures that the model can focus on the defect pattern without being affected by any external noise. In this example, the defect pattern type is "scratch," indicating that the defect tend to form elongated scratch patterns, which become more apparent after denoising. This technique helps the model to focus on the bad dice within the pattern and improve its ability.

C. Model Architecture

In recent years, Vision Transformer (ViT) has emerged as a promising deep learning model for vision tasks. MetaFormer has demonstrated that the success of ViT can be mainly attributed to its unique architecture. Based on this idea, we

propose the Convformer block and use it as the basic building block to construct our model. The model architecture are presented at the bottom of Fig. 1.

The ConvFormer block shown in the bottom left corner of Fig. 1 contains three modules. The first module is a 1×1 convolution layer that controls the number of feature maps. The second module is a 3×3 convolution layer, which functions as a token mixer to mix information between tokens. The third module is the channel MLP module. The last two modules adopt residual connection operation and apply group normalization [8] before themselves. Group normalization is an alternative to batch normalization. It divides input channels into groups and normalizes each group independently, reducing internal covariate shifts within the network and enhancing the model's stability and performance.

As illustrated at the bottom of Fig. 1, our model comprises eight blocks followed by two 1×1 convolution layers. This architecture allows us to mark the *SL* for multiple DUIs simultaneously at their original positions in the output image. We can then obtain these *SL*s by simply checking the values at the corresponding positions.

D. Training Strategy

In previous methods, DUI annotation was carried out on a single die. Each annotation would generate specific annotated data. Choosing a single die as DUI would lead to rapid data expansion. As the data dimensions increase and the amount of data grows, the training data would expand considerably, leading to excessively long training times. To address this issue, we have modified the training strategy by selecting multiple dice to be annotated as DUI at a time. This generates training data that effectively mitigates the problem of data expansion, thereby reducing the time required for training.

Increasing the number of dice selected for DUI annotation can reduce training time but may decrease visible information, affecting the model's ability to detect test escapes. It is crucial to find a balance between training time and model performance. Our approach performs best when 30 dice are annotated as DUIs simultaneously, allowing effective test escape detection while maintaining acceptable training time.

III. Metric and Setup for Evaluation

A. Evaluation Metric

For all subsequent experiments, we train the suspiciousness models using a common training set of wafers and evaluate their performance on a fixed test set. During evaluation, each die on a testing wafer takes turns to be the DUI to form an individual testing sample, where the die on its turn is annotated as the only DUI, and its test result is encoded as pass in the input-feature regardless of its actual result. The suspiciousness model is used to calculate the suspiciousness level of each testing sample. Any DUI with a suspiciousness level exceeding a given threshold will be discarded. Such setup on testing samples is to create the scenarios where an originally captured bad die as the DUI is somehow not detected by the current manufacturing tests and see whether the suspiciousness model can successfully identify this assumed test escape based on the test result of the other dice on the wafer. An effective model should identify as many test escapes as possible while minimizing the discard of original good cases to reduce yield loss.

In this paper, we adopt Equation 1 proposed in [3] to estimate the expectation of the number of actual test escapes covered by a GDBN method, denoted as CTE (covered test escapes), based on its captured assumed test escapes, denoted as NN (net negative), and the overall defect coverage of the applied manufacturing tests, denoted as DC.

$$CTE = \frac{(1 - DC)}{DC} \times NN \tag{1}$$

Then the profit of applying a GDBN method, denoted as *Gain*, can be calculated by Equation 2, where RMA represents how many times of a single die's selling price equals the penalty of receiving a customer return and $Loss$ represents the total number of good dice discarded by the GDBN method. As shown in Equation 2, the *Gain* defined here is the penalty to be paid minus the payment to be received if those test escapes are shipped to a customer, in terms of the times of a die's selling price. In other words, To maximize *Gain*, a GDBN method needs to ensure that the ratio of discarded good dice to covered test escapes is lower than the RMA of the targeted product, which can vary widely based on the product's application.

$$\begin{aligned} Gain &= RMA \times CTE - Loss \\ &= RMA \times \frac{(1 - DC)}{DC} \times NN - Loss \end{aligned} \tag{2}$$

In our later experiments, we use the *Gain* in Equation 2 as the metric to evaluate the effectiveness of GDBN methods. The values for DC and RMA are set to 99.9% and 500 by default. For more detailed explanations of the above equations, please refer to [3] or [4].

B. Overview of Evaluation Wafer Sets

In later experiments, all ML-based suspiciousness models use the same open wafer set called WM-811K [7] released by TSMC in 2015 for training and testing. The wafer set comprises 965 products totaling 811,457 wafers. Of these, 346 products are manually labeled for wafer defect patterns, with only 7.5% of the labeled wafers containing a meaningful defect pattern. Then, we selected 11 products, each containing a minimum of 500 wafers with image sizes smaller than 64$times$64 and meaningful defect-pattern labels. This dataset comprises 12,902 wafers categorized into 8 distinct defect-pattern types. A summary of these wafer sets can be found in Table I. For our study, 80% of these labeled wafers have been allocated for training, with the remaining 20% reserved for testing.

IV. Experiments Results

All experiments in this section are based on the testing set, which includes 2,581 labeled wafers with a meaningful defect pattern from the 11 products.

TABLE I
STATISTICS OF WAFER SET

Defect Pattern	# of Wafers	Ratio to Total Wafers	# of Dice	Yield
Center	2,706	21.0%	1,559,195	75.1%
Donut	141	1.1%	134,548	75.9%
Edge Local	2,288	17.7%	1,983,378	83.2%
Edge Ring	5,584	43.3%	10,396,877	85.4%
Local	1,411	10.9%	1,105,363	83.2%
Near-full	67	0.5%	48,356	13.5%
Scratch	383	3.0%	312,034	87.4%
Random	330	2.6%	315,318	49.8%
Total	12,902	100%	15,855,069	83.0%

A. Setting Different Suspiciousness-level Thresholds

The result of applying our GDBN method to the testing set with different thresholds on the predicted suspiciousness level for determining whether to discard a die is shown in Table II. Table II first lists the total number of test escapes TE in this wafer set, which is estimated by Equation 1 with NN substituted by the total number of bad dice in this wafer set. Then Table II lists the numbers of bad dice (NN) and good dice ($Loss$) that have an outputted suspiciousness level larger than a given threshold and will be discarded. Then, based on the NN, the corresponding covered test escapes (CTE) can be obtained with Equation 1. Also, by dividing $Loss$ by the total number of dice, the corresponding yield loss can be obtained. Next, the corresponding *Gain* can be calculated by Equation 2. In addition, DPPM can be calculated by dividing the remaining test escapes ($TE-CTE$) with the total number of dice. Last, the probability of hitting a bad die, $NN/(NN+Loss)$, is also listed in Table II. Note that the baseline in Table II represents the result if no GDBN method is applied, meaning that no good dice will be discarded.

As shown in Table II, when the suspiciousness-level threshold is higher, the corresponding ratio of NN versus $Loss$ becomes higher, meaning that the probability of capturing an assumed escape is also higher. When the suspiciousness-level threshold becomes lower, its NN and CTE can still increase but the increase of $Loss$ becomes more dramatic, and so is the decrease of the NN hit rate. This result demonstrates that a higher suspiciousness-level outputted by our suspiciousness model can indeed reflect a higher probability of a DUI being a test escape. Meanwhile, when moving to the next smaller suspiciousness-level threshold, the *Gain* can increase only when the new added $Loss$ versus the new added CTE is smaller than RMA to 1. The peak of the *Gain* in Table II occurs when the threshold is set to 0.7, where the CTE is 300, and the DPPM improvement is 114 (from original 205 to 91) while paying 0.31% of the yield as the cost.

Note that by setting a smaller step between adjacent thresholds in Table II, a *Gain* versus yield loss curve can be drawn. In later experiments, we will use such curves to compare the effectiveness of different GDBN methods.

B. Comparison against Previous Works

In the subsection, we compare our GDBN method with the conventional BNR GDBN and three other ML-based GDBN methods [1] [3] [4]. The conventional BNR GDBN uses simply the BNR (bad neighbor ratio) within the 3×3 window to reflect the suspiciousness level of a DUI. [1] uses bad neighbor counts of different regions within an observation window as input features and applies linear regression to train the suspiciousness model. [3] uses an 7×7 image as the input features to encode each die's pass/fail information within the observation window and applies a MLP to train the suspiciousness model. [4] uses a full-wafer image as the input features to encode each die's pass/fail information and the wafer-level defect pattern for each targeted wafer and applies MobileNet to train the suspiciousness model.

Fig. 3 plots the *Gain*-versus-yield-loss curve of each GDBN method in this comparison. As shown in Fig. 3, our curve is above any of the other curves with a significant margin, meaning that the *Gain* achieved by our GDBN method is always significantly higher than that of any other GDBN method when everyone gives up the same percentage of yield. This result also demonstrates that the suspiciousness level outputted by our suspiciousness model can better correlate the probability of a DUI being bad even though its test result somehow shows passed so that it can result in a higher NN hit rate, a larger CTE, and a lower $Loss$-versus-NN rate, which eventually lead to a higher *Gain*.

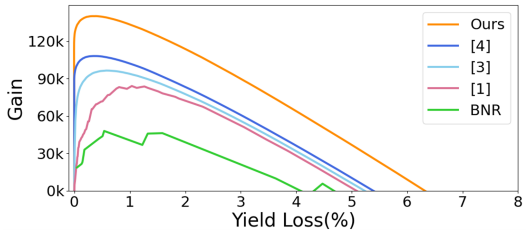

Fig. 3. Gain-versus-yield-loss curves for GDBN methods

In addition, the two suspiciousness models using a full-wafer image and the targeted wafer's defect pattern as input features, ours and [4], can consistently outperform the other models in Fig. 3, which demonstrates that a wafer-view defect pattern is a critical and effective information for identify a potential test escape. This result also reflects that using a local observation window, like conventional BNR GDBN, [1] and [3], is not enough to spot a potential test escape effectively. Furthermore, the conventional BNR GDBN method here performs worse than any of the listed ML-based suspiciousness models, showing the effectiveness of directly correlating the suspiciousness model to the pass/fail result of a DUI through supervised learning.

C. Defect-Pattern-Prominent Preprocessing

Table III lists the maximum *Gain* with and without applying our defect-pattern-prominent preprocessing technique to filter out scattering bad dice for the training wafers. As shown in Table III, the maximum *Gain* will drop from 140,813 to

TABLE II
DETAILED RESULT OF APPLYING OUR SUSPICIOUSNESS MODEL WITH DIFFERENT SUSPICIOUSNESS-LEVEL THRESHOLDS

Suspiciousness level (SL) Threshold	TE	Net Negative (NN)	Covered TE (CTE)	Loss	Yield Loss (YL)	Δ = new added #s to a higher *SL* Threshold					Gain	DPPM
						ΔNN	ΔCTE	ΔLoss	NN Hit Rate	$\frac{\Delta Loss}{\Delta CTE}$		
no GDBN	538	0	0	0	0	0	0	0	0	0	0	205
SL⩾0.9		268,311	269	1,479	0.05%	268,311	269	1,479	99.5%	5.5×	132,810	103
SL⩾0.8		284,790	285	4,615	0.15%	16,479	16	3,136	84.0%	196.0×	137,923	97
SL⩾0.7		299,319	300	9,768	0.31%	14,529	15	5,153	73.8%	343.5×	140,041	91
SL⩾0.6		314,820	315	18,697	0.59%	15,501	15	8,929	63.5%	595.3×	138,871	86

107,528. This 33,285 difference on the maximum *Gain* here shows the impact of applying our defect-pattern-prominent technique. This result also demonstrates that the detection of a potential test escape by our suspiciousness model really relies on the identification of a wafer's defect patterns and hence highlighting the defect patterns with a preprocessing technique can effectively help the training and guide suspiciousness model to perceive this concept.

TABLE III
IMPACT OF DEFECT-PATTERN-PROMINENT PREPROCESS

Method	No Preprocess	With Preprocess
Max. Gain	107,528	140,122

D. Multiple-DUI Training

When generating each training sample, multiple (m) dice are randomly selected simultaneously as the DUIs. As can be imagined, when m is larger, more information is excluded for determining the suspiciousness level of a DUI and hence the prediction problem becomes more complicated. However, when inferencing the suspiciousness model on a testing wafer, only one single die is selected as the DUI so that the model can have complete information to make a judgment. In other words, we set a more challenging training goal for the suspiciousness model than its actual application.

TABLE IV
TRAINING WITH (m)-DUI LABELING

# of Masked Dice	Max. Gain	Epochs	Avg. Training Time per Epoch	Total Training Time
1	136,906	17	7,249 sec	34.2 hr
5	137,747	30	1,416 sec	11.8 hr
10	139,582	24	714 sec	4.8 hr
30	140,122	52	243 sec	3.5 hr
50	134,061	22	168 sec	1.0 hr

Table IV lists the maximum *Gain*, total training time, total number of epochs, and average training time per epoch for each number of m used for training. As shown in Table IV, the maximum *Gain* increases slightly along with the increase of m until m reaches 30. This result first shows that setting a higher training goal with a larger m does make the model smarter. Meanwhile, m cannot be set too large, otherwise the given information will not be enough for the model to summarize the rule behind. In addition, when m is larger, the number of training samples created from a training wafer becomes less, meaning the training time per epoch can become shorter, which is another benefit of using a larger m. However, the speedup here is not linearly proportional to the number of m because the number of epochs to convergence may vary from case to case. The convergence condition is set to the fifth consecutive epoch with no improvement in the loss function.

V. CONCLUSION

In this paper, we introduce a new preprocessing method that enhances the wafer-level information in the encoded data. Additionally, we simultaneously annotate multiple dice as DUIs for training. This significantly reduces data expansion issues and decreases the training time required. Experiment results show that our method outperforms other GDBN methods. It captures and removes more *TE*s to reduce *DPPM* more effectively and achieve better performance in *Gain*.

REFERENCES

[1] R. Miller and W. Riordan, "Unit level predicted yield: A method of identifying high defect density die at wafer sort," in *Proceedings International Test Conference 2001 (Cat. No.01CH37260)*, Jan. 2001, pp. 1118–1127.

[2] C. Xanthopoulos, A. Neckermann, P. List, K.-P. Tschernay, P. Sarson, and Y. Makris, "Automated Die Inking through On-line Machine Learning," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Jul. 2019, pp. 27–32.

[3] C.-H. Yang, C.-H. Yen, T.-R. Wang, C.-T. Chen, M. Chern, Y.-Y. Chen, J.-N. Lee, S.-Y. Kao, K.-C. Wu, and M. C.-T. Chao, "Identifying Good-Dice-in-Bad-Neighborhoods Using Artificial Neural Networks," in *2021 IEEE 39th VLSI Test Symposium (VTS)*, Apr. 2021, pp. 1–7.

[4] C.-M. Liu, C.-H. Yen, S.-W. Lee, K.-C. Wu, and M. C.-T. Chao, "Enhancing Good-Die-in-Bad-Neighborhood Methodology with Wafer-Level Defect Pattern Information," in *2023 IEEE International Test Conference (ITC)*, Oct. 2023, pp. 357–366.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017.

[6] W. Yu, M. Luo, P. Zhou, C. Si, Y. Zhou, X. Wang, J. Feng, and S. Yan, "MetaFormer Is Actually What You Need for Vision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 819–10 829.

[7] M.-J. Wu, J.-S. R. Jang, and J.-L. Chen, "Wafer Map Failure Pattern Recognition and Similarity Ranking for Large-Scale Data Sets," *IEEE Transactions on Semiconductor Manufacturing*, vol. 28, no. 1, pp. 1–12, Feb. 2015.

[8] Y. Wu and K. He, "Group Normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.