Wafer Map Failure Pattern Classification Using Deep Learning

Jie Gong (jieg@stanford.edu), **Chen Lin** (chenlin1@stanford.edu)

Abstract

In this study, we worked on how to automate the wafer map failure pattern classification using deep learning computer vision methods, which is a hot topic in the semiconductor industry nowadays. We preprocessed the pre-existing public dataset using data normalization and augmentation, explored both simplified AlexNet and simplified VGG16 model architectures, performed hyperparameter tuning to optimize our model performance, and reported the quantitative and qualitative results of our two models. It turns out that our simplified versions of AlexNet and VGG16 models can both achieve high accuracy, precision and recall, with simplified VGG16 outperforming simplified AlexNet.

1. Introduction and Related Work

Wafer inspection is very important for increasing the yield of a micro/nano-fabrication process in the semiconductor industry. Based on different kinds of detected wafer map failure patterns, it is possible to figure out the root causes of various process issues [1-4]. The traditional visual recognition approach performed by an experienced person can be expensive and time-consuming. Therefore, investigating how to automate the wafer map failure pattern classification is interesting and valuable, which can remarkably enhance the wafer inspection efficiency in comparison with manual inspection [1-4]. Novel deep learning methods are proposed in our project to accurately identify various defect patterns on wafers. The input to our models is a normalized 1-channel wafer map image ($42 \times 42 \times 1$) with only one failure pattern from the 8 defect types ("Scratch", "Edge-Ring", "Edge-Loc", "Center", "Random", "Loc", "Near-full" and "Donut"). We then used both simplified AlexNet and simplified VGG16 models to output the predicted defect pattern of this wafer map.

There were a great many early studies investigating wafer map failure pattern recognition (WMFPR) [5-8]. However, because these methods are of low accuracy, they are not good enough for large-scale dataset analyses. Recently, a few research groups have processed large-scale wafer map datasets accurately and efficiently using some novel techniques (e.g. feature extraction [1-4] and defect clustering [2]) based on various traditional machine learning pattern recognition algorithms (e.g. support vector machines [1-2], k-nearest neighbors [9], and decision trees [3]). In order to further improve the accuracy and efficiency of analyzing large-scale wafer map datasets, we applied deep learning convolutional neural network (CNN) models in our project.

2. Dataset and Features

Based on the public WM-811K(LSWMD) dataset from Kaggle [10], it consists of 811457 wafer maps with information about the wafer map, number of dies, lot name, wafer index in each lot, training or test set label and failure pattern type. Although this dataset has already been divided into the training and testing sets by experts, we did not follow this existing data separation and defined our own training and test datasets. Ideally, each of the 47543 lots should have 25 wafer maps, leading to 1157325 total wafer maps. However, in reality, some wafer maps are somehow missing in some lots. More importantly, according to the failure type information analysis, only 25519 wafer maps have real defect patterns, while 147431 wafers have no defect patterns (labeled "none") and 638507 wafers have no labels. Since currently we are only interested in wafers with real failure patterns, the vast majority of the wafer maps have been removed and only those 25519 samples are useful to us.

One big issue for this dataset is that the dimensions of the wafer maps are not uniform. Based on the waferMap column information ("0", "1" and "2" represent the regions with no die, a normal die and a defective die respectively) from the dataset, the wafer map dimensions can be extracted and turn out to be not the same size. Data normalization was first performed via dividing this waferMap column data by 2, which converted the data type from int to float and therefore made the resizing process possible. In order to minimize the upsizing and downsizing errors after the image size unification, we calculated the weighted average dimensions of the 25519 wafer maps in the x and y directions and found that the optimal dimensions should be chosen as 42×42. Also, the resizing errors during data

transformation can potentially serve as regularization. Figure 1 shows the wafer map comparisons before and after resizing, which indicates that the defect pattern types are not modified after the data transformation.

The other big issue for this dataset is its highly imbalanced data distribution among the 8 failure pattern types. We applied two data augmentation strategies to solve this data imbalance problem, namely flipping and rotating. The image comparisons before and after data augmentation are shown in Figure 2(a), which confirms that the failure pattern type is visually unchanged after the data augmentation.

As for the data split, two different approaches were used. Approach 1 is first performing data augmentation on the normalized wafer maps and then dividing the resulting data into the training and test datasets based on a 7:3 ratio. Approach 2 is first splitting the normalized wafer maps into the training and test datasets according to a 7:3 ratio and then applying data augmentation only on the training dataset. Compared with Approach 1, Approach 2 has the advantage of excluding the data augmentation effect from the test performance results, resulting in potentially better evaluation of the model performance on the test dataset.

It can be seen from Figure 2(b) and Figure 2(c) that, for both Approach 1 and Approach 2, after data augmentation, the ratio of the maximum number of images for a certain failure pattern type ("Edge-Ring" in this case) to the minimum number of images for a particular defect pattern type ("Near-full" in this case) is less than 10. Compared with the highly imbalanced failure pattern type distributions before data augmentation (Figure 2(b) and Figure 2(c)), the data diversity is significantly increased, which is beneficial to the training processes.

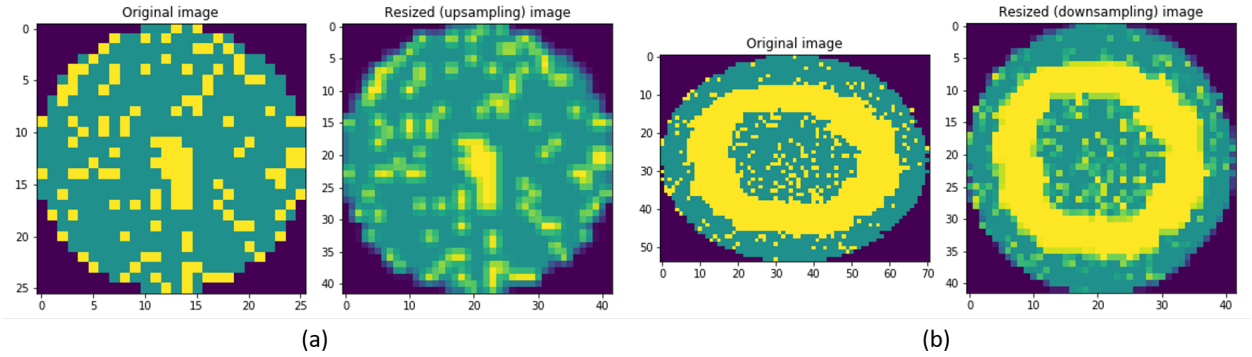


Figure 1. Wafer map comparisons before and after image (a) upsizing or (b) downsizing.

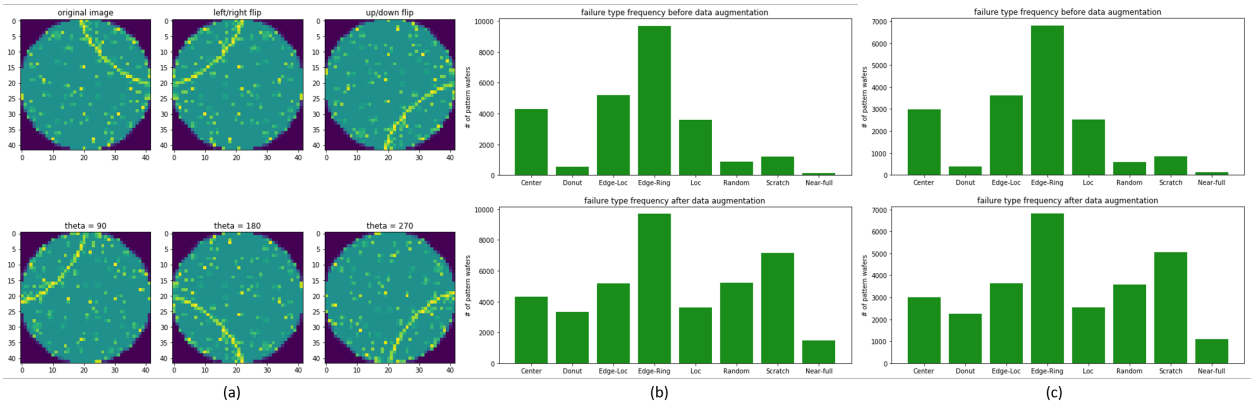


Figure 2. (a) Wafer map comparisons as well as failure pattern type distributions using (b) Approach 1 and (c) Approach 2 before and after data augmentation.

3. Methods

First, we built a simplified version of AlexNet CNN model [11] (Figure 3) by using the mini-batch gradient descent and Adam optimization. The justifications of this simplification can be summarized as follows. First, our input image size $42 \times 42 \times 1$ is much smaller than that in original AlexNet [11]. Moreover, the image contents used for

developing AlexNet are much more complex than ours[2]. Last, our goal is to classify only 8 defect pattern types while AlexNet is aimed at 1000 classes[2]. Therefore, we reduced the kernel size and the number of filters in our simplified AlexNet model to reduce time and computation costs.

Second, we built a simplified VGG16 CNN model [12] (Figure 4) on the basis of the mini-batch gradient descent and Adam optimization. Compared with the original VGG16 model, the number of layers, 16, does not change. However, the number of filters in each layer is remarkably reduced. The major reason we performed this simplification is because it is very difficult to make the original VGG16 model converge to a global minimal. In addition, the original VGG16 requires high computational resources and its training process is slow.

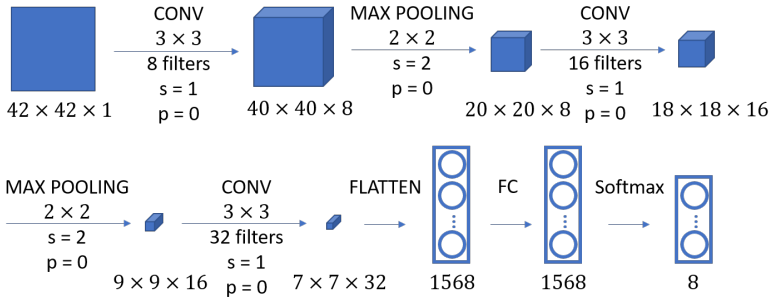


Figure 3. Our simplified AlexNet model architecture.

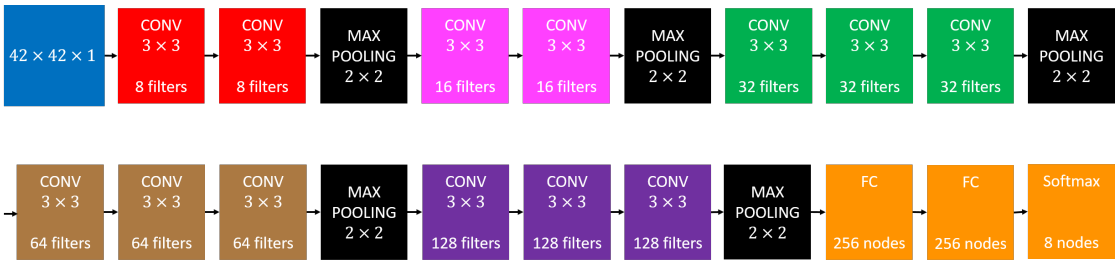


Figure 4. Our simplified VGG16 model architecture.

4. Experiments, Results and Discussion

TensorFlow 1.13 framework was used through the whole project. During the training processes of our models, we applied TensorBoard for neural network visualizing. As an example, Figure 5(a) and Figure 5(b) show our simplified AlexNet and simplified VGG16 model graphs generated by TensorBaord respectively. Also, for the purpose of easy error checking, we fixed the random seeding of data splitting, image shuffling and weight Xavier initialization. Furthermore, to avoid data overfitting, we investigated the effects of L2 regularization and dropout regularizations. Table 1 shows the overall training accuracy, testing accuracy and variance of our two models with and without regularization. In our simplified AlexNet case, using Approach 1 or Approach 2 yields very similar performance results. It can be seen from Table 1 that adding L2 regularization or dropout regularization can contribute to higher testing accuracy and lower training accuracy for our simplified AlexNet model, which can potentially prevent overfitting. Moreover, our simplified AlexNet model outperforms the conventional machine learning based benchmark on Kaggle (overall training accuracy: ~80.4%; overall testing accuracy: ~79.0%) [13]. In our simplified VGG16 case, when using Approach 1 with no regularization (Figure 5(c)), the simplified VGG16 model shows almost perfect training accuracy (~99.8%) and pretty high testing accuracy (~92.0%), which performs better than the simplified AlexNet model correspondingly. However, when using Approach 2 in the simplified VGG16 case (Figure 5(d)), the training process could not converge to a global minimal, resulting in accuracy close to random guessing. Figure 5(d) shows that the training process is stuck in some local minimal even after starting a couple of epochs, with its cost (~2) close to the initial cost ($\ln 8 \approx 2.08$). We attempted to change the weight initialization, learning rate, mini-batch size and optimizer parameter but none of them worked.

To see how our two models performed on each specific defect pattern class, the training and testing pattern recognition confusion matrices were generated under various regularization and hyperparameter conditions (Figure 6(a) and Figure 6(b) show 2 examples). It is easy to calculate the recall, precision and F1 score performance metrics based on a confusion matrix. Figure 6(c) shows these performance metrics for our simplified VGG16 model with no regularization as an example. As is show in Figure 6(a) and Figure 6(b), the "Loc" defect type has the lowest testing accuracy among all the 8 failure pattern classes. The misclassification of both our models can be partly explained by the wafer map normalization. Due to the resizing of image dimensions, the wafer maps become more blurred, which makes the defect pattern identification more difficult.

In order to further improve the performance of our simplified AlexNet and simplified VGG16 models, we performed the hyperparameter tuning of learning rate (0.001, 0.005 and 0.01), L2 regularization coefficient lambda (0.001, 0.01 and 0.1) and dropout regularization keep rate (0.5, 0.75 and 1) on the test dataset. We selected these hyperparameters for tuning because we believe they are important to our model performance results. There are totally 27 tests. Due to the space limitation, the detailed results are not listed here and only the most important trends are discussed as follows. First, based on the dropout regularization keep rate tuning results, larger keep rate leads to lower bias and higher variance, which is as expected. Therefore, 0.5 is chosen as the best keep rate for the dropout regularization. Figure 7 shows the scattering plot of the testing accuracy when the keep rate is kept as 0.5. It indicates that our model performance is most sensitive to lambda. When lambda is equal to 0.001, even setting learning rate to 0.1 gives decent results. However, if lambda equals 0.1, the testing accuracy still stays low even with a small learning rate.

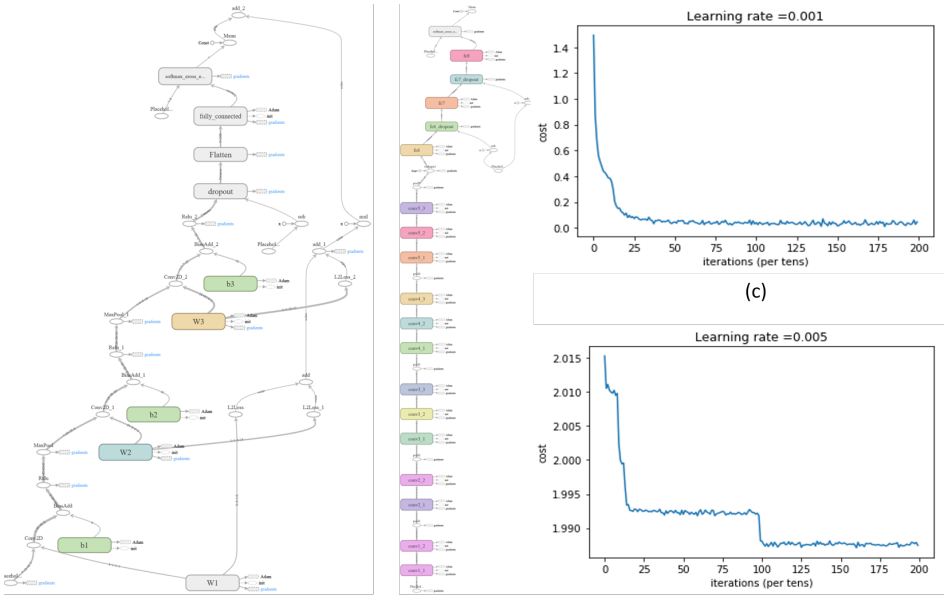
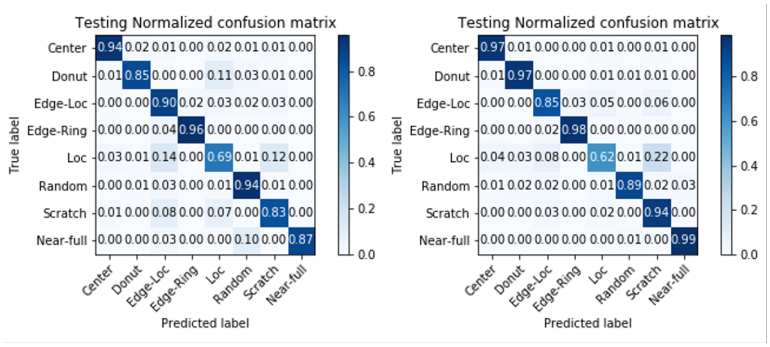


Figure 5. Our (a) simplified AlexNet and (b) simplified VGG16 model graphs generated by TensorBoard, as well as cost vs. iterations plots for our simplified VGG16 model using (c) Approach 1 with dropout regularization (keep rate = 0.5) and (d) Approach 2 with no regularization.

Table 1. Bias and variance metrics of all the models.

Model	Data Split Approach and Regularization	Training Accuracy	Testing Accuracy	Variance
Simplified AlexNet	Approach 1 (no regularization)	96.5%	88.5%	8.1%
	Approach 1 (L2 regularization with lambda = 0.001)	93.8%	88.9%	4.9%
	Approach 1 with L2 and dropout regularizations (lambda = 0.001 and keep rate = 0.5)	92.6%	90.6%	2.0%
	Approach 2 (no regularization)	98.5%	88.2%	10.4%
	Approach 2 with L2 regularization	96.5%	89.2%	7.3%
	Approach 2 with L2 regularization plus dropout	92.0%	89.7%	2.3%
Simplified VGG16	Approach 1 (no regularization)	99.8%	92.0%	7.8%
	Approach 1 with dropout regularization (keep rate = 0.5)	98.3%	91.5%	6.8%
	Approach 2 (no regularization)	Similar to random guessing accuracy due to no convergence to a global minimal		
	Approach 2 with dropout regularization (keep rate = 0.5)			



Defect Type	Recall	Precision	F1 Score
Center	96.60%	94.30%	95.40%
Donut	96.50%	92.20%	94.30%
Edge-Local	85.50%	85.10%	85.30%
Edge-Ring	97.70%	97.90%	97.80%
Local	62.00%	79.50%	69.60%
Random	89.50%	97.40%	93.30%
Scatch	94.10%	85.10%	89.30%
Near-full	98.90%	91.00%	94.80%

(a) (b) (c)

Figure 6. Testing normalized confusion matrices of (a) our simplified AlexNet model with L2 regularization and dropout regularization (lambda = 0.001 and keep rate = 0.5) as well as (b) our simplified VGG16 model with dropout regularization (keep rate = 0.5). (c) Recall, precision and F1 score performance metrics for our simplified VGG16 model with no regularization.

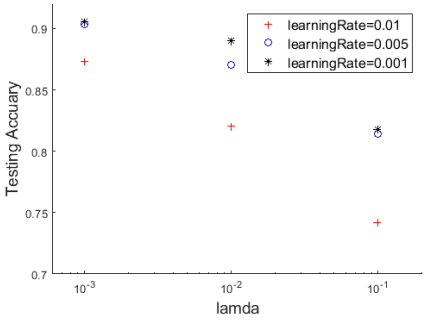


Figure 7. Testing accuracy results for hyperparameter tuning (keep rate = 0.5).

5. Conclusion and Future Work

In this work, we implemented simplified AlexNet and simplified VGG16 CNNs for automating wafer map defect pattern classification with high training and testing performance. For future work, we would like to solve the non-convergence issue of the simplified VGG16 model using Approach 2 and compare it with other cases. Another research direction which is worth exploring is to use transfer learning to classify wafer map failure patterns. Although VGG16 is computationally expensive, we can freeze the front layers of a pre-trained VGG16 model and only train the last 1 or 2 layers of the neural networks.

6. Contributions

Jie Gong and Chen Lin worked together to investigate the research background, to develop, optimize and analyze the models, as well as to document the project. Jie Gong focused more on data processing and model implementation while Chen Lin concentrated more on literature review and project documentation.

7. GitHub Repository

https://github.com/gongjie437/CS230-Deep-Learning

8. Acknowledgements

We thank our project teaching assistant Zahra Koochak for her useful feedback on our project.

References

[1] Wu, Ming-Ju, Jyh-Shing R. Jang, and Jui-Long Chen. "Wafer map failure pattern recognition and similarity ranking for large-scale data sets." *IEEE Transactions on Semiconductor Manufacturing* 28.1 (2014): 1-12.

[2] Fan, Mengying, Qin Wang, and Ben van der Waal. "Wafer defect patterns recognition based on OPTICS and multi-label classification." *2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. IEEE, 2016.

[3] Piao, Minghao, et al. "Decision Tree Ensemble-Based Wafer Map Failure Pattern Recognition Based on Radon Transform-Based Features." *IEEE Transactions on Semiconductor Manufacturing* 31.2 (2018): 250-257.

[4] Yu, Jianbo, and Xiaolei Lu. "Wafer map defect detection and recognition using joint local and nonlocal linear discriminant analysis." *IEEE Transactions on Semiconductor Manufacturing* 29.1 (2015): 33-43.

[5] Wang, Chih-Hsuan. "Recognition of semiconductor defect patterns using spatial filtering and spectral clustering." *Expert Systems with Applications* 34.3 (2008): 1914-1923.

[6] Yuan, Tao, Suk Joo Bae, and Jong In Park. "Bayesian spatial defect pattern recognition in semiconductor fabrication using support vector clustering." *The International Journal of Advanced Manufacturing Technology* 51.5-8 (2010): 671-683.

[7] Hwang, Jung Yoon, and Way Kuo. "Model-based clustering for integrated circuit yield enhancement." *European Journal of Operational Research* 178.1 (2007): 143-153.

[8] Yuan, Tao, and Way Kuo. "A model-based clustering approach to the recognition of the spatial defect patterns produced during semiconductor fabrication." *IIE Transactions* 40.2 (2007): 93-101.

[9] Kim, Byunghoon, et al. "A regularized singular value decomposition-based approach for failure pattern classification on fail bit map in a DRAM wafer." *IEEE Transactions on Semiconductor Manufacturing* 28.1 (2015): 41-49.

[10] https://www.kaggle.com/qingyi/wm811k-wafer-map

[11] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

[12] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[13] https://www.kaggle.com/ashishpatel26/wm-811k-wafermap