A Reliable Defect Detection Method for Patterned Wafer Image Using Convolutional Neural Networks with the Transfer Learning

Ji-Hee Lee¹ and Jee-Hyong Lee²*

¹Department of Semiconductor Display Engineering, Sungkyunkwan University, South Korea
²Department of Computer Science and Engineering, Sungkyunkwan University, South Korea

*Corresponding author: john@skku.edu

Abstract. In the semiconductor manufacturing process, wafer inspection images have valuable information on defects and wafer yield. It is worthy to analyze these images and there are many algorithms for this. Whenever new product designs are introduced, the algorithms are used to detect defects. Most new product designs have new types of defects. Hence, it is important to update the algorithms to cope with new types of defects. To update the algorithms, engineers collect the data and adjust parameter values, such as threshold, to detect defects. It is time-consuming to collect enough data and only a knowledgeable engineer can find an appropriate parameter value. However, there is always a lack of engineer resources and time in the manufacturing industry. Due to these reasons, many algorithms can’t be used reliably and have disappeared. Therefore, we propose the advanced method to update an algorithm easily using a deep learning model. Deep learning models achieve high accuracy with less engineer knowledge than traditional algorithms. But it also needs a lot of data to train the model, so we apply an advanced method of updating with a small amount of data in a short time. This is called transfer learning. Once we train a deep learning model with high accuracy, we can update the model with a small amount of data. Transfer learning uses the information gained during the training. As a result, we make a model that is easy to update with high accuracy. In the experiments, we obtained 99% accuracy with sufficient training data. To update the model for the new data, we did transfer learning and got 97% accuracy. It was lower, but we only used 10% of the training data and 5% of the training time. Also, the accuracy we have obtained is enough to be used for defect analysis. Our approach outperformed and executed faster than traditional algorithms.

1. Introduction
In the semiconductor industry, detecting defect is an important issue because it is directly connected to the productivity and the revenue. There are eight different major semiconductor manufacturing processes and those processes are repeated until the wafer becomes a product. As a result, a wafer goes through several tens of manufacturing processes which could make defects. In order to reduce the overall process time, the wafers are not inspected in every process except when the inspection doesn’t increase.
the process time. It is hard to detect defects with a limited inspection data. In addition, whenever a new type of wafer pattern with a new type of defect is introduced, an algorithm update is inevitable.

To update the algorithm, it is necessary to have enough data to analyze the defects and time to adjust the parameter values, such as threshold, to detect defects. However, collecting enough data is not easy. Although every wafer has their inspection data, these are not labelled data. They need to be labelled manually and it is very time consuming. In addition, they are highly imbalanced data. Normal data is always many times more than the defect data. Imbalanced data generally leads a wrong result. Most of the stable manufacturing processes have the same problem. Adjusting the parameter values are also difficult. Only knowledgeable engineers can decide the parameter value to detect.

There are a number of studies to detect and classify defects. Shankar and Zhong [1] showed how to find defects and visualize them. Tsai and Yang [2], Zontak and Cohen [3] also described how to compare two images and detect defects. These approaches used the golden image to detect a defect using a colour difference. Thus, the golden image should be prepared whenever a wafer needs to be inspected. Pak et al. [4] explained how to automatically set the threshold that separates objects and a background based on the grey-scale distribution. This algorithm is not easy to be applied to a patterned wafer. One of the famous algorithms is die-to-die comparison which is used commonly [5]. To compare dies, it needs to find a die boundary and to set a threshold. It requires a lot of engineer effort and is vulnerable to an ambiguous boundary. In addition to these algorithms, some machine learning based approaches are used for the defect classification [6], [7], [8]. Those approaches achieve high accuracy, but they need a lot of data to train.

All of the above-mentioned studies require a lot of time and effort from the engineers. However, there are limited engineer resources and time in the manufacturing industry. To overcome these difficulties, we propose the advanced method to detect defects using a deep learning model. The deep learning models we use is based on the Convolutional Neural Network (CNN) such as VGG16 [9], VGG19 [9] and Inception v3 [10] which are specialized for the image processing and a feature extraction. These models achieve high accuracy and require less engineer knowledge to detect defects. One disadvantage of these models is that they require a lot of training data. This is not suitable for frequent updating of a new type of wafer pattern. So, we apply the transfer learning [11], [12] to update the models easily. The transfer learning is a method of retraining a trained model for additional data. Once we train the model with sufficient data for the accuracy at the first time, we can update the model with small additional data in a short time without losing accuracy. Thus, we don’t need to collect much data for updating and are able to fast respond. This is a great benefit to the industry and these benefits enable us to obtain a reliable model by updating the model as needed.

This paper is organized as follows. Section 2 states the problem defines. We explain the method and model architecture we used in section 3. Next, section 4 describes experiments and section 5 provides the conclusion.

2. Proposed method

In order to improve the detection accuracy, we adopt the Convolutional Neural Network (CNN) that shows the best result for the image processing. CNN uses a convolution layer instead of a fully connected layer. This convolution layer extracts features from the image easily. All of the images we use vary in complex patterns and colours depending on the manufacturing process. So, we use CNN that is specialized in extracting the features. At the beginning of this experiment, we evaluated a simple CNN models that have 3 to 7 layers and we got only 50% accuracy. Therefore, we apply three more complicate models including VGG16 [9], VGG19 [9] and Inception v3 [10]. We evaluate these models in aspects to accuracy and re-trainability. VGG16 and VGG19 are a simpler model than Inception v3. Only difference between VGG16 and VGG19 is a layer depth. Inception v3 is wider and deeper but it has fewer parameter than VGG16 and VGG19.

After training using these models, we use the transfer learning method. The transfer learning is proper to update the models with small effort. Once we train the models to have high accuracy, we can update the models for additional data. If the updated model doesn’t have sufficient accuracy, we can notice it
early since the transfer learning performs fast. During the transfer learning, we freeze some blocks as shown in Figure 1 and 2. By freezing the blocks, we can reuse the knowledge from the trained model. We only train some unfrozen blocks with additional data. Most of the blocks share the knowledge and some blocks learn the special feature from additional data. In this paper we call this retraining process as a fine-tuning.

2.1. Network architecture
VGG16, VGG19, Inception v3 are already proved for the image classification problem. We use the ImageNet weight for the training. After these models, we attach a dropout layer, global average pooling layer [13] and a dense layer. The dropout layer makes the model generalize. The global average pooling reduces the number of parameters in the model prior to the classification. Even if we use images of different size, we can still use the same parameter size using this layer before the classification layer. The last dense layer prepares a data structure for classification. We use a binary classifier because we only judge whether defect exists or not. We depict the number of blocks to freeze for fine-tuning in Figure 1 and 2.

2.1.1. VGGNet. VGG16 and VGG19 are generally called VGGNet that introduced the 2014 ILSVRC. Depends on the layer count, it is named VGG16 or VGG19. The most powerful point of these models is simplicity. They use only 3 x 3 convolution layer and max pooling. These models just stack the layers and right most is the top. We evaluate the performance depends on the layer count using these two models. VGG19 has more three layers than VGG16. Figure 1 shows the model architecture.

2.1.2. Inception v3. As seen in Figure 2, Inception v3 is a deeper and wider model than VGGNet. It is also introduced the 2014 ILSVRC and won. This model is composed of inception module that includes convolution layers and pooling layer in parallel. Inception module makes the model trainable even though deep and wide. We mark the inception module as a block. Despite of a wider and deeper model, it has fewer parameter than VGGNet. So, this model converges very quickly.

2.2. Input transformation
The result image is 2048x2048 with a low resolution. This size is too large to train therefore we rescale the images to 512x512. We use the Keras library to train. This library provides several image transformation options. We apply four transformation including rescale, shear, zoom and horizontal flip. Table 1 shows each transformation setting type and value. These transformations are randomly applied to images when it loaded.

### Table 1. Data transformation type

| Transformation type | value |
|---------------------|-------|
| Rescale             | 0 ~ 1 |
| Shear               | 0.2   |
| Zoom                | 0.2   |
| Horizontal Flip     | True  |

3. Experiments

As mentioned above, we use three famous models to detect defects. Keras library we use provides the trained model using ImageNet dataset. So, we first fine-tune the provided models with the transfer learning for additional data. And then, we train the models with our training images. This is called the full training. During the full training, the models are taught the features of the wafer images. Finally, we fine-tune the trained models through the full training with transfer learning. We evaluate all these experiments in terms of accuracy and re-trainability. Building a reliable system requires the model that can be retrained quickly with high accuracy. We focus on this point during the experiments.

3.1. Dataset

The dataset we use is an image of inspection for CMP process from Samsung which is semiconductor company in South Korea. Once a wafer is processed, an inspection image is taken. We get these images and make a database. We only use the partial data generated in October 2018. The training data is ten types of wafer pattern. All types classified by the original method are highly imbalanced. We extract same count by the label to balance the data. After that, we modify some labels because the original method misclassifies some data. It makes a little imbalance. Table 2 provides a summary of the data. The error rate means the percentage of the misclassified label by the original method. Among ten types, we train and validate using seven types (A to G) of images. Remained three types (H to J) of images are used for fine-tuning and test. Aside from these images, we add additional images to check the performance of a deep learning model. Added additional images are not distinguished by the original method because of the ambiguous boundary or colour variation. We call these images as ‘Unclassified’.

### Table 2. Data description

| Purpose       | Type | Pass | Fail | Total | Error rate |
|---------------|------|------|------|-------|------------|
| Training &    | A    | 267  | 361  | 628   | 7.50%      |
| Validation    | B    | 699  | 996  | 1695  | 11.30%     |
|                | C    | 486  | 586  | 1072  | 4.70%      |
|                | D    | 1863 | 2137 | 4000  | 5.90%      |
|                | E    | 295  | 391  | 686   | 7.00%      |
|                | F    | 461  | 703  | 1164  | 10.40%     |
|                | G    | 1873 | 2127 | 4000  | 3.40%      |
| Test &        | H    | 218  | 189  | 407   | 27.30%     |
| Fine-tune     | I    | 212  | 240  | 452   | 13.10%     |
3.2. **Experimental setup**

All experiments are performed on a 2.50GHz Intel Xeon with 256GB memory and NVDIA Quadro M6000 with 24GB Memory. We use the pre-trained models provided from Keras library. It has already been proven that pre-trained models achieve better performance. The pre-trained models are trained using ImageNet [14]. ImageNet is large scale image database for deep learning.

3.3. **Fine-tuning with pre-trained model**

All models are pre-trained using ImageNet and already used for fine-tuning in other area. So, we fine-tune and then validate these models without any extra training. In order to fine-tune and validate the models, we use three test data (Type H, I, J). The total number of images we gather is approximately 4000, but we only use 1000 training and 500 validation samples to balance the type. Each type is included at the same rate. We train during 50 epochs with 5 batch size. We load the pre-trained models and freeze some bottom layers to fine-tune. We experiment on three conditions to evaluate how many layers are effective for fine-tuning. All training models are composed by the blocks, so we freeze using this block unit. We sequentially evaluate by freezing top two blocks, one block and only last fully-connected layer. Table 3 shows the accuracy and F1-score. VGGNet achieves around 87% accuracy but Inception v3 has maximum 70% accuracy. VGGNet is better than Inception v3 but all models are under 90%. These pre-trained models provided from Keras are trained by the large scale of images. However, those images include things that can be seen everywhere like cat and dog. They are different from the wafer image with repeated patterns. And even though a wafer pattern is same, the colour of image looks differently depends on process. Therefore, the frozen bottom blocks of three pre-trained models using ImageNet are more likely to have less relevant features with a wafer. Improper knowledge leads to low model accuracy.

| Unfrozen layer | VGG16     | VGG19     | Inception v3 |
|----------------|-----------|-----------|--------------|
|                | Accuracy  | F1 Score  | Accuracy     | F1 Score  | Accuracy     | F1 Score  |
| Last layer     | 81.40%    | 0.82      | 81.00%       | 0.81      | **70.00%**   | 0.7       |
| 1 block        | 87.00%    | 0.87      | 85.20%       | 0.85      | 63.00%       | 0.61      |
| 2 blocks       | **87.40%**| 0.87      | **87.00%**   | 0.87      | 64.00%       | 0.62      |

3.4. **Full training**

To get more relevant bottom module with a wafer image, we train all models using seven types (A to G). Among the images, we randomly select 10,000 samples for training and 3,000 samples for validation. We train the models with 100 epochs and 5 batch size. All models achieve high accuracy. It is better performance than the original method which is about 93%. Figure 3 (a), Figure 3 (b) show the convergence speed is little different. Inception v3 is the fastest and VGG19 followed. The complicate model is generally slower than the simple model and it is harder to train. However, Inception v3 has high accuracy, fast convergence speed and short training time. Table 4 explains the full training result and shows that Inception v3 is the best in all aspects. In addition, the training time of VGG16 and VGG19 shows if the model is same, the deeper model needs more time.
Figure 3. (a) Accuracy (b) Loss trend depends on models

Table 4. Full training results using data A to G in the dataset.

| Model     | Training time (hour) | Training Accuracy | Training Loss | Validation Accuracy | Validation Loss | F1-Score |
|-----------|----------------------|-------------------|--------------|---------------------|-----------------|----------|
| VGG16     | 41.2                 | 98%               | 0.064        | 99%                 | 0.038           | 0.99     |
| VGG19     | 46.6                 | 98%               | 0.050        | 99%                 | 0.033           | 0.99     |
| Inception v3 | 36                  | 99%               | 0.036        | 99%                 | 0.025           | 0.99     |

In Table 5, we evaluate two data types. One is ‘Unclassified’ data we mentioned, and the other is additional data.

Table 5. Evaluation Result using data H to J and ‘Unclassified’ in the dataset.

| Data Type          | VGG16 | VGG19 | Inception v3 | Original method |
|--------------------|-------|-------|--------------|-----------------|
|                    | Accuracy | F1 Score | Accuracy | F1 Score | Accuracy | F1 Score | Accuracy |
| Unclassified       | 90%     | 0.90   | 91%         | 0.91   | 91%     | 0.91     | Can’t evaluate |
| Test (H, I, J)     | 73%     | 0.71   | 68%         | 0.65   | 80%     | 0.8      | 84%      |

‘Unclassified’ data that accounts for 25% of the total. Original method can’t detect the defects but all models we trained get about 90% accuracy. This accuracy is not bad but a little lower than the training result. That’s because these models don’t have information about the ambiguous boundary.

In the industry area, reliability is as important as high accuracy. Therefore, we evaluate the additional data. The additional data are similar to the training data in terms of wafer images. However, the result is not satisfactory. It is worse than original method and also previous fine-tuning result. Even though the bottom module is similar, final high features like pattern are different from training data. It may lead to the poor result.

Through this experiment, we find that the deep learning models are appropriate to detect defects in terms of accuracy. But in other respects, it’s a little different. Although ‘Unclassified’ data achieves the accuracy about 90%, additional data are not close to 90%. Both evaluation results show that only full training can’t cover the new type of data. In addition, training time needs to be considered. In this experiment, the shortest time of training is 36 hours. In the manufacturing industry it is not acceptable that the model training takes more than one day since it means defective wafers are produced during one
day without engineer’s notice. This one-day training time doesn’t even include the pre-processing time for the new training dataset.

3.5. Transfer Learning with full training weights

According to the previous experiments, the full training is also not enough to make a reliable detecting system. We need a training for additional data as short as possible. The transfer learning, we mentioned section 2, is useful to train for small amount of data in a short time. Based on the previous training result, we train additional data with freezing bottom blocks. The previous training weights contain useful low features in the bottom module. These features help the models get better accuracy. This experiment is done in the same way and same condition as section 3.3.

All models spend about 2 hours that is 5% of the previous training time and used 10% of the full training data. Table 6 shows the transfer learning result. VGG16 and VGG19 are achieved maximum 97% accuracy however Inception v3 is just 90%. All models are the best performance when training the models with unfrozen top two blocks. We also evaluate the ‘Unclassified’ in best experimental conditions. An original method can’t evaluate at all, but we can detect defects using our method even when transfer learning. Table 7 shows the evaluation result. It is not bad accuracy considering that it is not trained data.

Figure 4 shows the accuracy and loss chart. We only draw the best performance in a chart. Contrary to the full training result, Figure 4 show that VGG19 converges fast in this experiment. When only the last layer is frozen, all models show noticeably low performance. We add three blocks condition in this experiment. Using this condition, we check the effect of the increase in the number of the frozen layers. When too many blocks are frozen, models can’t train the high features to be used to detect. Less freeze the blocks, it may cause over-fitting. It also leads to poor performance.

Table 6. Transfer learning result using data H to J.

| Unfrozen layer | VGG16 Accuracy | F1 Score | VGG19 Accuracy | F1 Score | Inception v3 Accuracy | F1 Score |
|----------------|----------------|----------|----------------|----------|------------------------|----------|
| Last layer     | 57.60%         | 0.49     | 67.60%         | 0.65     | 83.80%                 | 0.84     |
| 1 block        | 96.80%         | 0.97     | 92.60%         | 0.93     | 87.60%                 | 0.88     |
| 2 blocks       | **97.00%**     | 0.97     | **97.00%**     | 0.97     | **90.00%**             | 0.9      |
| 3 blocks       | 87.00%         | 0.87     | 87.00%         | 0.87     | 65.00%                 | 0.6      |

Figure 4. (a) Accuracy (b) Loss trend depending on models.
Table 7. Transfer learning evaluation using ‘Unclassified’ data.

| Unfrozen layer | VGG16        | VGG19        | Inception v3 |
|---------------|--------------|--------------|--------------|
|               | Accuracy     | F1 Score     | Accuracy     | F1 Score     | Accuracy     | F1 Score     |
| 2 blocks      | 90.22%       | 0.90         | 86.11%       | 0.86         | 89.22%       | 0.89         |

Through this experiment, we find that transfer learning is suitable for fast respond. Once we train the models with high accuracy, we can update the models with small amount of data in a short time. It is slightly lower than the full training, but it enables to analyze the defects fast. This makes the models reliable.

4. Conclusion

In the manufacturing industry, there are limited engineer resources and time. With this limitation, we detect defects whenever a new type of wafer pattern is introduced. To cope with this problem, we propose the advanced method to update using a deep learning model. Deep learning models we use are specialized for the image processing. It leads to high accuracy models to detect defects. And we can detect defects that can’t be detected by an original method. However, these models need a lot of data for a training. So, we apply the method to update with small amount of data in a short time. This is called transfer learning. This reduces the burden to collect data. It is great beneficial in the industry area. Using our proposal, we achieve 99% accuracy in the full training. Then, we obtained 97% accuracy when transfer learning for a new type of data. This is slightly lower than the full training, but it is sufficient for fast respond to analyze the defects. This approach enables us to obtain a reliable model by updating the model as needed.

As the process and the design pattern become more complicated, it is important to not only judge whether defects exist but also classify defects. Therefore, this approach needs to be improved to classify the detailed types of defects. We remain this as a future work.

References

[1] NG Shankar, ZW Zhong. Microelectronic engineering 77: p. 337-346 (2005)
[2] DM Tsai and RH Yang. Image and Vision Computing 23: p. 1094-1101 (2005)
[3] M Zontak and I Cohen. Machine Vision and Applications 21: 129-141 (2010)
[4] M Pak, J Park and S Kim. Computer Science and its Applications. Springer, Berlin, Heidelberg. p. 1085-1091 (2015)
[5] I Isomura. U.S. Patent No. 7,630,535. (2009)
[6] KB Lee, S Cheon and CO Kim. IEEE Transactions on Semiconductor Manufacturing 30: p. 135-142 (2017)
[7] T Nakazawa and DV Kulkarni. IEEE Transactions on Semiconductor Manufacturing 31: p. 309-314 (2018)
[8] G Tello, OY Al-Jarrah, PD Yoo, Y Al-Hammadi, S Muhaidat, and U Lee. IEEE Transactions on Semiconductor Manufacturing 31: p. 315-322 (2018)
[9] K Simonyan and A Zisserman. arXiv preprint arXiv:1409.1556 (2014).
[10] C Szegedy, W Liu, Y Jia, P Sermanet, S Reed, D Anguelov, D Erhan, V Vanhoucke and A Rabinovich.IEEE. computer vision and pattern recognition (2015)
[11] SJ Pan and Q Yang. IEEE Transactions on knowledge and data engineering 22: p. 1345-1359 (2010)
[12] L Shao, F Zhu and X Li. IEEE transactions on neural networks and learning systems 26: p. 1019-1034 (2015)
[13] M Lin, Q Chen and S Yan arXiv preprint arXiv:1312.4400 (2013)
[14] J Deng, W Dong, R Socher, LJ Li, K Li and L Fei-Fei. "Imagenet: A large-scale hierarchical image database.". p. 248-255 (2009)