공학박사학위논문

Intelligent Data Acquisition for Predictive Modeling in Manufacturing Systems

제조 시스템에서의 예측 모델링을 위한 지능적 데이터 획득

2021 년 2 월

서울대학교 대학원

산업공학과

심 재 웅

Intelligent Data Acquisition for Predictive Modeling in Manufacturing Systems

제조 시스템에서의 예측 모델링을 위한 지능적 데이터 획득

지도교수 조 성 준

이 논문을 공학박사 학위논문으로 제출함

2020 년 12 월

서울대학교 대학원

산업공학과

심 재 웅

심재웅의 공학박사 학위논문을 인준함

2021 년 1 월

위 원 장	박 종 헌	(인)
부위원장	조 성 준	(인)
위 원	이 재 욱	(인)
위 원	강 필 성	(인)
위 원	김 동 일	(인)

Abstract

Intelligent Data Acquisition for Predictive Modeling in Manufacturing Systems

Jaewoong Shim
Department of Industrial Engineering
The Graduate School
Seoul National University

Predictive modeling is a type of supervised learning to find the functional relationship between the input variables and the output variable. Predictive modeling is used in various aspects in manufacturing systems, such as automation of visual inspection, prediction of faulty products, and result estimation of expensive inspection. To build a high-performance predictive model, it is essential to secure high quality data. However, in manufacturing systems, it is practically impossible to acquire enough data of all kinds that are needed for the predictive modeling. There are three main difficulties in the data acquisition in manufacturing systems. First, labeled data always comes with a cost. In many problems, labeling must be done by experienced engineers, which is costly. Second, due to the inspection cost, not all inspections can be performed on all products. Because of time and monetary constraints in the manufacturing system, it is impossible to obtain all the desired inspection results. Third, changes in the manufacturing environment make data acquisition difficult. A change in the manufacturing environment causes a change in the distribution of

generated data, making it impossible to obtain enough consistent data. Then, the model have to be trained with a small amount of data. In this dissertation, we overcome this difficulties in data acquisition through active learning, active feature-value acquisition, and domain adaptation. First, we propose an active learning framework to solve the high labeling cost of the wafer map pattern classification. This makes it possible to achieve higher performance with a lower labeling cost. Moreover, the cost efficiency is further improved by incorporating the cluster-level annotation into active learning. For the inspection cost for fault prediction problem, we propose a active inspection framework. By selecting products to undergo high-cost inspection with the novel uncertainty estimation method, high performance can be obtained with low inspection cost. To solve the recipe transition problem that frequently occurs in faulty wafer prediction in semiconductor manufacturing, a domain adaptation methods are used. Through sequential application of unsupervised domain adaptation and semi-supervised domain adaptation, performance degradation due to recipe transition is minimized. Through experiments on real-world data, it was demonstrated that the proposed methodologies can overcome the data acquisition problems in the manufacturing systems and improve the performance of the predictive models.

Keywords: Predictive modeling, Data acquisition, Manufacturing systems, Active learning, Active feature-value acquisition, Domain adaptation, Wafer map pattern classification, Fault prediction, Fault detection and classification model

Student Number: 2018-39406

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Predictive modeling is to find the functional relationship $y = f(X)$ between the input variable X and the output variable y. This corresponds to the supervised learning, where the output variable to be predicted is directly utilized for training, and can be divided into classification task and regression task. In classification task, an output variable to be predicted has categorical values, and regression is a case where an output variable to be predicted has continuous values.

The process of predictive modeling is divided into a training phase and an prediction phase. The training phase requires a labeled training dataset that contains a large number of pairs of input variable values and output variable values. The functional relationship $y = f(X)$ is inferred from the dataset using a learning algorithm. The constructed prediction model f is used in the prediction phase. In prediction phase, given a new data point x that does not have the corresponding value of output variable, the prediction model estimates the y value of that point. In this way, predictive modeling is used to predict the value of the output variable with the value of the input variable. The general process of predictive modeling is illustrated in Figure 1.1.

Predictive modeling through machine learning approaches is widely adopted in

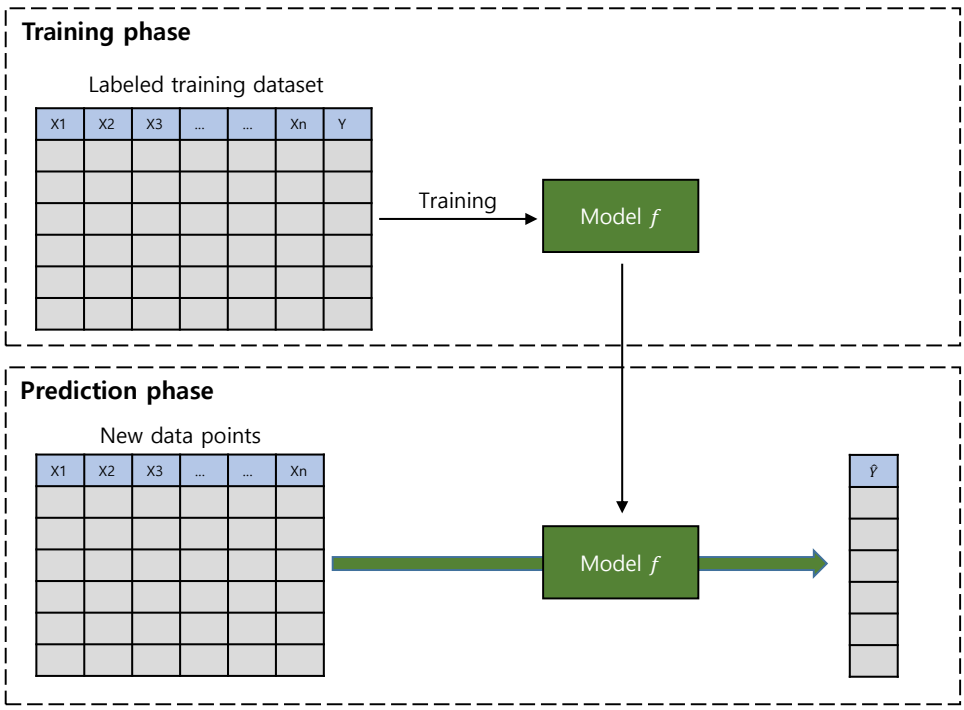


Figure 1.1: General process of predictive modeling.

manufacturing systems [35, 1, 49, 78]. First of all, a representative example is the automation of visual inspection [16, 10]. A lot of manufacturing process steps involve visual inspection procedures by engineers to find product defects. Considerable research has been conducted to automate the visual inspection by constructing a model using the inspection history data of the engineer as training data. Predicting faulty product is also an example of using predictive modeling in manufacturing systems [90, 39, 36, 40, 38]. If the faults are predicted in the early stage of the processes and appropriate action can be taken, unnecessary costs can be reduced. A variety of related information for a product is used to constitute input variables for the fault prediction. This includes sensor measurements, inspection results, environmental

factors, process parameters, and process time. Output variables indicate whether a product will be found faulty in the future, such as outgoing inspection fault or customer-found fault. In addition, predictive modeling can also be used to replace expensive inspection process [11, 37]. Various inspections are performed during the manufacturing process, some of which are very expensive. Virtual metrology in semiconductor manufacturing is a typical example. By constructing a predictive model based on the various types of sensor data as input and physical metrology result as output, physical metrology process can be partially replaced, thereby reducing cost.

To build a high-performance predictive model, it is essential to acquire high quality data. However, in manufacturing systems, it is practically impossible to obtain enough data of all kinds that are needed for the predictive modeling. There are three main difficulties in the data acquisition in manufacturing systems. First, labeled data always comes with a cost. In many problems, labeling must be done by experienced engineers, which is costly. Second, acquiring inspection result data incurs inspection costs and time delays. Because there are time and monetary constraints in the manufacturing system, it is impossible to obtain all the desired inspection results. So, in the case of high-cost inspection, only some sampled products can be inspected. Third, changes in the manufacturing environment make data acquisition difficult. The manufacturing environment is not fixed and constantly changes. A change in the manufacturing environment causes a change in the distribution of generated data, making it impossible to obtain enough consistent data. Then, the performance of an existing predictive model is degraded, and the model needs to be retrained with new data. In order to retrain a model, data generated in a new environment is sufficiently required, but there is little data at the beginning after

the change, making model training difficult.

We overcome this data acquisition problems in manufacturing systems through active learning, active feature-value acquisition, and domain adaptation. Active learning is a framework to construct the high-performance model with the reduced label acquisition cost. In active learning, prediction model is built using active querying, by which instances are labeled for training. The main idea of active learning is to selectively label the most informative instances for the model. The informative instances are selected from unlabeled instances and queried to be labeled. Then, the model is constructed with the labeled instances. The procedure is repeated to obtain high prediction performance without labeling the entire instances. Whereas active learning focuses on the cost of acquiring the value of an output variable, the active feature-value acquisition focuses on the cost of acquiring the value of input variables. Obtaining feature-values is as expensive as obtaining labels in real-world manufacturing processes. Thus, active feature-value acquisition is as essential as active learning in practice. Domain adaptation is a methods to utilize labeled data in a relevant source domain for a target domain to be solved. It can be used when the data of the target domain is insufficient. The key idea is how to transform a large amount of information in the source domain to fit the target domain. A schematic explanations of these three methodologies are shown in Figure 1.2.

The predictive modelings we target in each chapter are the wafer map pattern classification and fault prediction problems. Wafer map pattern classification in semiconductor manufacturing is to classify defect types of the wafer maps through the pattern of them. The wafer map consist of spatial information of whether each die is pass or fail, and it shows how the fail dies are distributed on the wafer. The defect

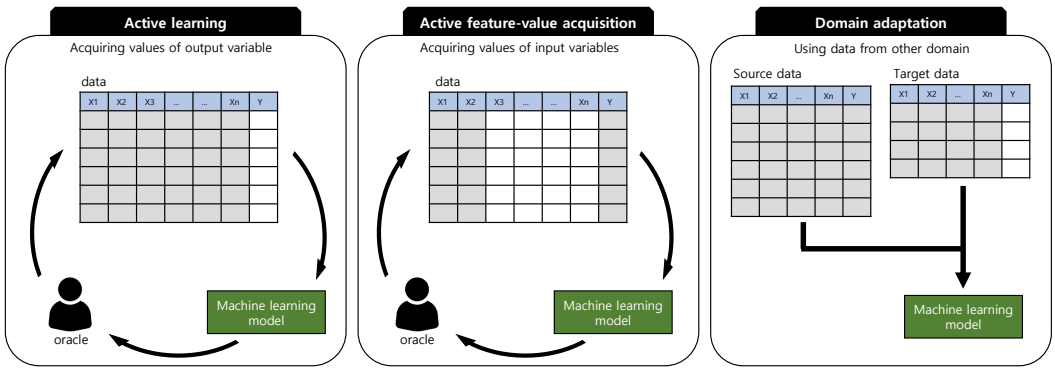


Figure 1.2: Illustrative descriptions of active learning, active feature-value acquisition, and domain adaptation.

type of the wafer map provides important information for determining the defect cause as wafer maps with similar patterns are likely to have the same root cause for a defect. To construct a high-performance classification model, a large amount of labeled training data is required. However, it is expensive to secure a large amount of labeled wafer maps, as the wafer maps must be annotated by expert engineers. We solve this problem using an active learning strategy.

Fault prediction is to predict in advance whether each product will be defective in the future. When trying to perform fault prediction with inspection results as input, inspection cost becomes an obstacle for predictive modeling. Due to cost constraints, not all products can go through all inspections, and some expensive inspections are conducted as sampling inspections. If this sampling is performed intelligently to effectively obtain the value of input variables, the performance of fault prediction can be improved. Active feature-value acquisition is employed to obtain high performance with reduced inspection cost.

Fault prediction can also be performed with time series sensor data generated

during the process. In particular, in the semiconductor manufacturing process, this kind of model is referred to as a fault detection and classification (FDC) model. In each process step, a manufacturing recipe is set. When this recipe is changed, the distribution of the collected time series sensor data also changes. Therefore, whenever a recipe transition occurs, it is necessary to retrain the FDC model with the new recipe data, but it takes time to secure a lot of labeled training data for the new recipe. For this problem, domain adaptation is utilized to reduce the performance degradation that occurs in recipe transition situations and quickly build a new recipe model.

As described above, in this dissertation, predictive modelings in the semiconductor manufacturing process were used as the main case for demonstrating each methodology. The semiconductor manufacturing process is largely divided into fabrication, probe test, assembly, and final test. The role of predictive modeling covered in each chapter in the entire process is shown in Figure 1.3. The model that predicts the result of the probe test with sensor data of the fabrication process is dealt with in chapter 6, the automation of wafer map pattern classification in the probe test is dealt with in chapter 3 and 4, and the model that predicts the final test result with the measured values in the probe test is dealt with in chapter 5. Table 1.1 summarizes each predictive modeling, difficulty in data acquisition, and strategy.

The reminder of this dissertation is organized as follows. In chapter 2, literature reviews on the related three methodologies and predictive modeling problems in manufacturing systems are presented. chapter 3 proposes an cost-effective wafer map pattern classification system based on active learning of a convolutional neural network (CNN) to address the high labeling cost. In chapter 4, the above active

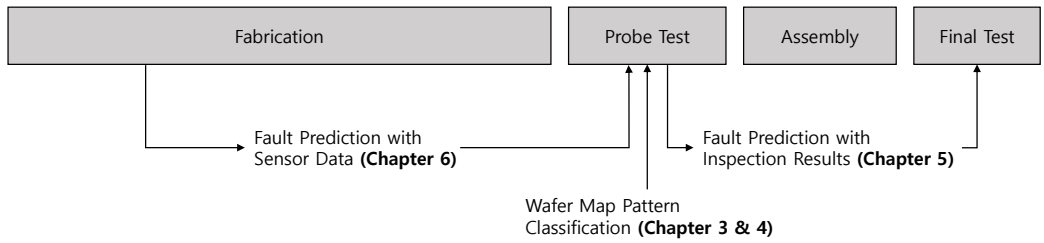


Figure 1.3: The composition of the semiconductor manufacturing process and the relative position of each predictive modeling.

learning framework for the wafer map pattern classification is improved based on the cluster-level annotation. In chapter 5, we propose an active inspection framework in which products are not randomly sampled for advanced inspections and are instead intelligently sampled with prediction uncertainty to achieve high accuracy while maintaining low inspection costs. chapter 6 presents the adaptive fault detection framework to minimize the performance degradation caused by the transition of recipe. Finally, chapter 7 provides the conclusions and discussion, as well as the directions for future work.

Table 1.1: Outlook of the dissertation.

Chapter	**Difficulty in data acquisition**	**Strategy**	**Target Problem**
Chapter 3 & 4	Labeling cost	Active learning	Wafer map pattern classification
Chapter 5	Inspection cost	Active feature-value acquisition	Fault prediction with inspection results
Chapter 6	Recipe transition	Domain adaptation	Fault detection and classification with time series sensor data

Chapter 2

Literature Review

2.1 Review of Related Methodologies

2.1.1 Active Learning

Active learning is a case of machine learning, in which a prediction model is built using active querying, by which instances are labeled for training. This method is used when it is difficult to label all instances because of a high labeling cost. In an active learning framework, informative instances are selected to be labeled by the labeler.

Uncertainty sampling is the most basic and traditional strategy, and queries the most uncertain instances from among the unlabeled set. An easy method for estimating uncertainty is to utilize the posterior probability of a predicted class [48]. Settles *et al.* [74] proposed to query instances with the least confident instance. The margin between the highest posterior probability and the second highest probability was considered as the uncertainty in Scheffer *et al.* [70]. The entropy of the class posterior probabilities was exploited as an uncertainty measure by Hwa [30].

Some studies have also considered diversity of selected instances in uncertainty sampling methods because redundant instances add little information for updating the model. In Brinker [8], a diversity constraint was introduced for active learning

with support vector machine (SVM). Yang *et al.* [89] proposed a method of applying diversity maximization constraints to a multi-class problem. Xu *et al.* [88] selected the cluster centers of the instances lying within the margin of a SVM. Hoi *et al.* [28] chose instances that reduce Fisher information. Azimi *et al.* [2] used a variant of a Gaussian Mixture Model to maintain diversity among selected instances.

Query-by-committee [75] is another well-known technique which employs multiple prediction models. The prediction is performed for all unlabeled instances and the instance that is the most inconsistent between prediction models is queried. Other methods include the method of expected error reduction [66], which estimates the expected error after a set of queries to find the optimal query instances, and the total expected variance minimization method [74, 73], which reduces the generalization error indirectly.

Recently, with the development of deep learning, active learning algorithms for deep neural networks have also been appearing. However, they are not prevalent compared to the algorithms for traditional machine learning [22]. Most of the methods are based on uncertainty sampling because they can compensate for the high computational cost of deep neural networks. Simple uncertainty sampling methods using posterior probability have been successfully applied to deep neural networks [84]. In applying Bayesian methods to deep learning, Gal *et al.* [21] showed an equivalence between a dropout and the approximate Bayesian inference. Thus, it was possible to estimate uncertainty through multiple forward passes with Monte Carlo (MC) dropout [22].

In active learning for deep neural networks, there have been studies to take the diversity of the selected instances into account. Sener *et al.* [72] proposed a

distribution-based method for choosing instances representing the distribution of the entire unlabeled pool in an intermediate feature space of a deep neural network. But it has a high computational cost because it requires solving mixed integer programming. Kirsch *et al.* [42] proposed batchBALD method that increase diversity of selected instances with approximation of mutual information. BatchBALD recently reported superior performance in batch mode active learning tasks.

Additionally, cluster-based active learning was proposed. In the conventional active learning framework, annotation is conducted for each instance. Meanwhile, in cluster-based active learning, annotation is conducted for each cluster [63]. Therefore, clustering is performed on unlabeled instances. In this method, multiple instances in a cluster are equally annotated as the most frequent class in one annotation task. Cluster-based active learning assumes that the annotation cost for a cluster and an instance are the same. With this assumption, cluster-based active learning can yield a larger number of labeled training instances with the same annotation cost compared with the conventional active learning framework.

In Chapter 2 and 3, our active learning system uses a CNN for a classification model. We focus on uncertainty sampling owing to its simplicity and computational efficiency. Furthermore, we consider the diversity of the selected instances in order to solve the class imbalance problem.

2.1.2 Active Feature-value Acquisition

Active feature-value acquisition is a framework improve the trade-off between model performance and the feature-value acquisition cost. Obtaining feature-values is as expensive as obtaining labels in real-world manufacturing processes. Thus, active

feature-value acquisition is as essential as active learning in practice.

Active feature-value acquisition has been studied in two research directions, each of which focuses on the training phase and prediction phase, respectively.

Active feature-value acquisition for the training phase is to train the high performance prediction model while reducing the feature-value acquisition cost. Labels are given for all training instances, but the feature-values of some instances are missing. In this situation, in a similar manner to active learning, the most informative instances for the prediction model are selected and queried to acquire their feature-values. Through iteration of the feature-value acquisition of selected instances and model retraining, prediction performance of the model increases cost-effectively. The active feature-value acquisition for the training phase has been actively studied. Acquisition based on variance of imputed data (AVID) is a method to obtain values of instances with high imputation uncertainty via multiple imputation. Goal-oriented data acquisition (GODA) is a method to obtain values for instances that maximize model performance with imputation [92]. Melville *et al.* [56] proposed a method to obtain values of input variables to maximize the predefined utility function. Sankaranarayanan *et al.* [68] proposed a method to obtain values of instances based on the likelihood to be classified correctly. Dhurandhar *et al.* [15] selected the instances based on a derived upper bound on the expectation of the distance between the next classifier and final classifier.

The trade-off between model performance and the feature-value acquisition cost occurs not only in the training phase, but also in the prediction phase. If a new instance to be predicted has all the feature-values, the prediction for the instance would be accurate. However, in practice, it may not be possible due to the acquisi-

tion cost, meaning that some feature-values for the instance can be missing. Active feature-value acquisition for the prediction phase is to improve the prediction for the instance with minimal acquisition of feature-values. Cost-sensitive decision trees (DTs) [51] and cost-sensitive naïve Bayes [9] are constructed through learning algorithms designed to minimize the feature-value acquisition cost in prediction time. In [32], the Hidden Markov Model was used to train the sequence of feature-value acquisition. Desjardins *et al.* [14] used a cascaded ensemble of classifiers to obtain feature-values until the confidence threshold is met. All these works have focused on selecting features to acquire for each instance. On the other hand, few studies have focused on selecting instances to acquire a set of features, which can be called instance-completion setting. A study for the instance-completion was conducted in [34], which selected customers to acquire more information for cost-effective customer targeting.

The active inspection framework proposed in Chapter 5 corresponds to active feature-value acquisition in the prediction phase, especially in the instance-completion setting. Inspired by GODA [92] which is an imputation-based method for training phase, we also propose expected prediction change (EPC), a new active feature-value acquisition method. EPC can improve the performance of the active inspection framework. The missing value imputation task in EPC is conducted with a regression model. This is referred to as virtual metrology [11] because missing values correspond to skipped inspections [37].

2.1.3 Domain Adaptation

In order to obtain high performance using a deep learning model, a large amount of labeled training data is required. However, in many real-world problems, unlabeled data is readily available, while labeled data is often difficult to be obtained [3, 59]. Domain adaptation methods have been developed in order to utilize labeled data in a relevant source domain to a new target domain with little labeled data. There are differences in data distribution between different domains. To solve this problem, most domain adaptation methods try to find a feature space that can align the source representation and the target representation by using a parallel structure such as siamese networks [13].

The most basic approach for domain adaptation is a discrepancy-based approach. This approach measures the discrepancy between feature distributions of source and target domain and trains a model to minimize it. In many studies, Maximum Mean Discrepancy (MMD) [5] has been adopted as a discrepancy measure. MMD computes the norm of difference between the mean values of distributions of source and target domain. In Deep Domain Confusion (DDC) [80], MMD loss was added to a regular classification loss for the source domain to learn a domain invariant representation. Deep Adaptation Network (DAN) [53] used a multiple kernel variant of MMD which is defined as the reproducing kernel Hilbert space (RKHS) distance between the mean embeddings of different distributions.

Another typical approach is a classifier-based approach. This approach uses a domain classifier in order to learn a representation that is simultaneously discriminative of source labels while not being able to distinguish between domains. [79] introduced a domain classifier, a binary classifier that separates domains, and trained a model

through adversarial learning to find a representation that could not distinguish each domain. [23] directly maximized the loss of the domain classifier by reversing its gradients instead of using adversarial learning.

In Chapter 6, we use a discrepancy-based approach with MMD loss because it can be used for both an unsupervised adaptation setting and a supervised adaptation setting, so it can be easily extended to semi-supervised settings [85]. With this approach, we propose a framework that can effectively build an FDC model in a recipe transition situation.

2.2 Review of Predictive Modelings in Manufacturing

2.2.1 Wafer Map Pattern Classification

Numerous studies have been conducted on wafer map pattern classification. Most studies have focused on how to extract good features from wafer maps to classify their patterns. Wu *et al.* [86] performed wafer map classification through a SVM by extracting geometry-based and radon-based features. They released the WM-811K dataset used in the experiment for Chapter 3 and 4. Jeong *et al.* [31] developed a methodology using a spatial correlogram to find spatial autocorrelation, and used it to classify wafer map patterns. Yu *et al.* [91] proposed a method for extracting useful information from various features by using joint local and nonlocal linear discriminant analysis (JLNDA). Piao *et al.* [64] proposed a decision tree ensemble learning methodology using radon-based features. Fan *et al.* [18] performed multi-label classification of a wafer map using the "Ordering Point to Identify the Cluster Structure" (OPTICS) approach and a SVM. Saqlain *et al.* [69] proposed a voting ensemble classifier based on density, geometry, and radon-based features.

With advances in deep learning, CNNs have been actively applied to the semiconductor manufacturing process. Unlike traditional machine learning methods, a CNN enables automatic extraction of meaningful features from raw inputs without manual feature engineering. For the reason, many recent studies have demonstrated the effectiveness of CNN on learning from various types of data produced in the manufacturing process. Lee *et al.* [47] used a CNN to extract useful features for fault detection. Kim *et al.* [40] proposed a self-attentive CNN to detect faults from variable-length sensor data. As for the wafer map pattern classification problem, Nakazawa *et al.* [57] built a CNN model with synthetic wafer maps and applied it to the classification of actual wafer maps. Kyeong *et al.* [44] proposed constructing an individual CNN model for each defect type to classify wafer maps with mixed-type defect patterns. Nakazawa *et al.* [58] employed convolutional autoencoder to detect unseen wafer map pattern. Wang *et al.* [83] adopted adversarial learning to improve a CNN to better address imbalanced distribution of wafer map patterns. The research mentioned so far aimed to improve the performance of wafer map pattern classification through CNN. However, there has been little effort to reduce labeling cost that necessarily comes with deep learning. In Chapter 3 and 4, we addresses efficient labeling of wafers based on a CNN-based active learning system.

2.2.2 Fault Detection and Classification

Predictive modeling through machine learning approaches is widely adopted in manufacturing systems [35, 1, 49, 78]. Among many related problems, fault prediction for each product has been a representative application of predictive modeling [90, 39, 36, 40]. In fault prediction, a variety of related information for a product is

used to constitute input variables for fault prediction. This includes sensor measurements, inspection results, environmental factors, process parameters, and process time. Output variables indicate whether a product will be found faulty in the future, such as outgoing inspection fault or customer-found fault. Various learning algorithms have been employed for fault prediction, including logistic regression, decision tree (DT), support vector machine, artificial neural network, and ensemble-based model.

In particular, in the semiconductor manufacturing process, fault prediction model with time series sensor data generated during the process is referred to as a fault detection and classification (FDC) model. An FDC model is a model to predict each wafer's inspection results (faulty or normal) by using sensor data at a specific process step as input. Traditionally, studies on the FDC model used manually-extracted features as input instead of raw sensor data. Summary statistics of a sensor or combination of several sensors are usually adopted as input variables [61]. He and Wang [26] used principal component analysis for extracting features to reduce the computational complexity. In Park *et al.* [62], spline regression is applied and its coefficients are considered as the input features of SVM. One-class SVM has been also considered as a learning algorithm for fault detection model [54]. In studies using k-nearest neighbor (kNN) model [25], Mahalanobis distance was utilized as a distance measure [81], and dimension reduction through random projection was also used [93]. In addition, decision tree-based methods have also been studied [19, 12, 27]. However, these manually-extracted features would lose some important information contained in the raw sensor data, which may lead to performance degradation of the FDC model.

Recently, with the advance of deep learning models, raw sensor data is directly utilized as input of the models. Deep learning models have a feature extraction process internally, so they can handle the raw inputs and achieve high performance by reducing information loss. Lee *et al.* [46] developed an FDC model that is robust to noise by using stacked denoising autoencoder. Lee *et al.* [47] proposed an FDC model based on a CNN, which made it possible to identify the cause of failure without specialized knowledge. In Kim *et al.* [40], they proposed a self-attentive CNN in which self-attention mechanism is combined with CNN, which solved the problem that the time length of sensor data is different for each wafer. In the experiments of Chapter 6, we adopted the self-attentive CNN, the most recent model, as the FDC model used in the proposed framework. As far as we have investigated, there have been no studies for constructing the FDC model considering the transition of recipe.

Chapter 3

Active Learning for Wafer Map Pattern Classification

3.1 Problem Description

In semiconductor manufacturing, a wafer undergoes hundreds of complex fabrication steps, resulting in a large number of dies being created on the wafer [82]. Subsequently, each die on the wafer is subjected to various types of electrical tests. These tests determine whether each die performs as intended. According to the test results, each die can be represented by a binary value, *i.e.*, 0 for a pass, and 1 for a fail. The binary values on the wafer constitute the wafer map, *i.e.*, the wafer map shows the spatial distribution of the defective dies on the wafer.

Wafer maps can be classified into several defect types based on their spatial defect patterns. The defect type of the wafer map provides important information for determining the defect cause as wafer maps with similar patterns are likely to have the same root cause for a defect [52]. For example, a scratch-type defect is likely to be caused by particles and the hardening of the pad during chemical-mechanical planarization. A local-type defect is likely caused by non-uniformity or uneven cleaning [44]. If a certain defect type occurs within a certain period, the candidate for the cause of the defect can be ascertained based on the defect type.

Hence, engineers can further analyze and improve the process. Therefore, wafer map pattern classification (WMPC) is crucial in determining the cause of defects and improving the process to further enhance manufacturing quality.

With this necessity and importance, there has been considerable research on wafer map pattern classification. Especially recently, deep learning-based research has been actively carried out [57, 44]. A CNN is a typical deep learning-based methodology used in wafer map pattern classification. One requirement for the success of a deep learning methodology is a sufficient amount of labeled training data. However, it is too costly for experienced engineers to carry out manual labeling for the enormous number of wafers produced in the manufacturing process. It is even more difficult to obtain a sufficient number of wafer maps with spatial failure patterns, as most wafers do not have many failure dies, or correspond to non-pattern wafers. If we randomly sample wafers to be labeled, most wafers may correspond to a non-pattern wafer or may have spatial patterns similar to already-labeled ones. Thus, they would not be informative for improving the model. It is necessary to selectively annotate novel pattern wafers that have not yet been incorporated into the training set to improve the efficiency of the labeling.

In this chapter, we propose an cost-effective wafer map pattern classification system based on active learning of a CNN to address the above-mentioned problems. In the proposed system, there are four major steps: uncertainty estimation, query wafer selection, query wafer labeling, and model update. The CNN model for classifying wafer maps is trained by using an initial labeled set. With this model, the uncertainty of prediction in the unlabeled wafer maps is calculated in the uncertainty estimation step. Based on this uncertainty, the wafers to be labeled are selected in

the query wafer selection step. In the query wafer labeling step, the selected wafers are inspected by the engineer, and are merged into the labeled set. The CNN model is updated with the labeled set in the model update step. Through the repetition of these four steps, the performance of the CNN model is gradually increased. This method is cost-effective, because it can achieve higher performance with a lower labeling cost. Several methods for uncertainty estimation and query wafer selection are compared in this framework. The effectiveness of the proposed system is investigated through experiments using real-world data from a semiconductor manufacturer.

The main contributions of this chapter are summarized as follows. Firstly, we implement an efficient active learning system to build a CNN model with lower labeling cost for wafer map pattern classification. Secondly, diversified top-K selection method is proposed for the query wafer selection step that can alleviate class imbalance problem. Thirdly, we explore various uncertainty estimation methods to find the most suitable one for wafer map pattern classification.

3.2 Proposed Method

3.2.1 System overview

In this section, we introduce the entire structure of the proposed wafer map pattern classification system. The goals of this system are to reduce the amount of wafer maps inspected directly by engineers, and to build a classifier that achieves high classification performance with a small amount of labeled wafer maps. Figure 3.1 illustrates an overview of the proposed system. The system consists of four main steps: uncertainty estimation, query wafer selection, query wafer labeling, and model update.

At the start of this system, and in view of the large number of wafer maps that are not yet classified, the engineer randomly selects and inspects wafer maps manually. These randomly selected and labeled wafer maps constitute the training set of the initial model. We construct the initial CNN model with this small number of wafer maps. This CNN model performs a prediction on the unlabeled wafer maps, and also estimates the uncertainty of the prediction (uncertainty estimation). Based on these uncertainty values, the system selects wafer maps to be labeled by the engineer (query wafer selection). The engineer inspects the selected wafer maps (query wafer labeling). Newly-labeled wafer maps are integrated with existing labeled wafer maps, and the CNN model is updated with these labeled set (model update). Through repetition of this process, the CNN model is continually updated using the informative wafer maps, and eventually a high-performance classification model can be obtained. Because the engineer only inspects and labels a small number of selected wafers in each repetition phase, this process is more cost-effective than inspecting all of the wafers.

The notations for explaining the proposed system are presented in Table 3.1. Algorithm 1 presents the pseudocode for the entire process. The following subsections describe the four main steps.

Table 3.1: Notations used in Chapter 3.

Notation	Type	Description
$\mathbf{X}_i$	matrix	i-th wafer map in $\mathcal{D}$
y_i	scalar	label for $\mathbf{X}_i$
$\mathcal{D}$	set	whole wafer map dataset $\mathcal{D} = \{\mathbf{X}_i\}_{i=1}^{N}$
$\mathcal{D}^L$	set	labeled subset of $\mathcal{D}$.
$\mathcal{D}^U$	set	unlabeled subset of $\mathcal{D}$
K	scalar	query size for each repetition phase
T	scalar	number of iterations of forward passes for MC dropout
$\mathcal{W}$	set	CNN parameters
$\mathcal{W}_t$	set	randomly dropped CNN parameters in $\mathcal{W}$ for t-th iteration for MC dropout

Algorithm 1 Pseudocode for proposed system

Input: Dataset $\mathcal{D}$, query size per repetition phase K
Output: CNN parameters $\mathcal{W}$

procedure
  $\mathcal{D}^L \leftarrow$ random sample from $\mathcal{D}$
  Label each wafer map in $\mathcal{D}^L$
  $\mathcal{D}^U \leftarrow \mathcal{D} \setminus \mathcal{D}^L$
  Initialize $\mathcal{W}$ with $\mathcal{D}^L$
  while not reach maximum phase **do**
    Estimate uncertainty for each wafer map in $\mathcal{D}^U$
    $Q \leftarrow K$ most uncertain wafer maps from $\mathcal{D}^U$
    Label each wafer map in Q
    Fine-tune $\mathcal{W}$ with $\mathcal{D}^L \cup Q$
    $\mathcal{D}^L \leftarrow \mathcal{D}^L \cup Q$, $\mathcal{D}^U \leftarrow \mathcal{D}^U \setminus Q$
  end while
  return $\mathcal{W}$
end procedure

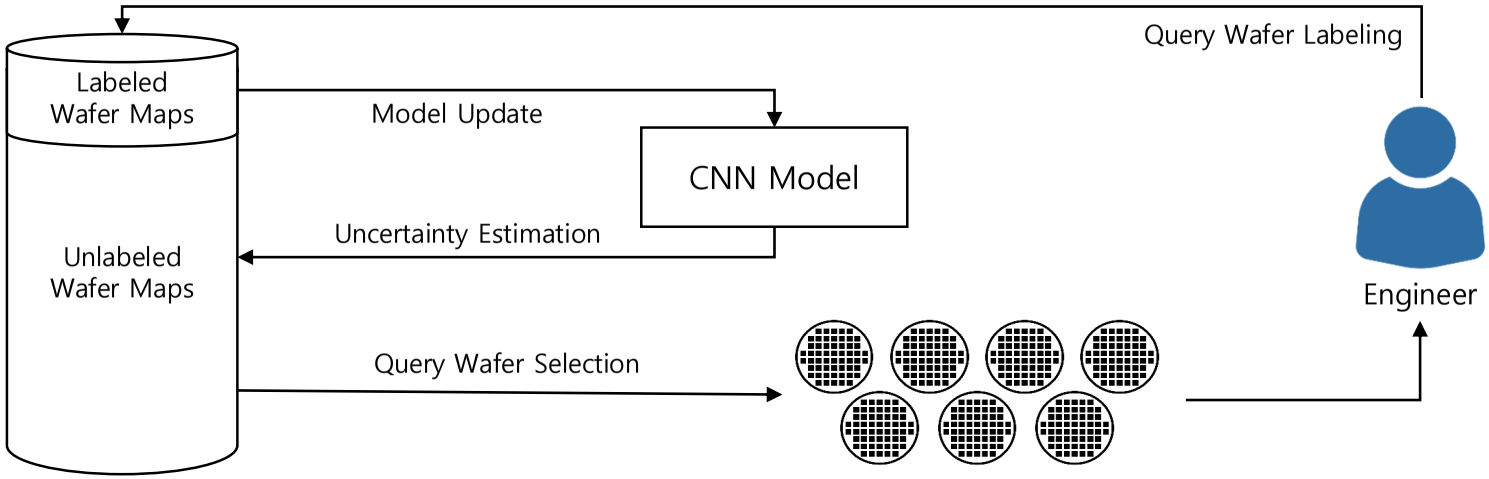


Figure 3.1: Overview of the proposed wafer map pattern classification system.

3.2.2 Prediction model

We adopt a CNN as a classification model for use in this system. The CNN is one of the representative methodologies of deep learning and has achieved state-of-the-art performance in various applications, especially in image classification tasks. Many studies have attempted to classify wafer map patterns using CNNs, and have achieved remarkable performance.

A CNN consists of convolution layers, pooling layers, and fully-connected layers. Usually, the convolution layer is used to extract features from the input data. The pooling layer serves to reduce the dimensions of the input data. Classification is performed in the fully-connected layers, using the features extracted from the convolution and pooling layers. With this CNN architecture, there is no need to execute extra feature engineering.

It is possible to employ an existing CNN model whose architecture has been optimized for model performance. As will be discussed in Section 3.2.3, if a Bayesian approximation is used, the model architecture should include a dropout.

3.2.3 Uncertainty estimation

A representative method for estimating the uncertainty of prediction by the CNN model involves using the output value of the last softmax layer. As the softmax output values can be interpreted as probabilities of belonging to individual classes, there are several methods that use these values. Typically, the uncertainty is quantified by least confidence, margin, and entropy measures.

All equations in this section represent the uncertainty of prediction for the i-th wafer. We will use the following equation to simplify the equations used for

uncertainty estimation. t represents each forward pass for MC dropout and has an integer value from 1 to T, *i.e.*, $t \in \{1, \ldots, T\}$.

$$\begin{aligned} p_j(y_i|\mathbf{X}_i) &= p(y_i = j|\mathbf{X}_i; \mathcal{W}) \\ p_j^t(y_i|\mathbf{X}_i) &= p(y_i = j|\mathbf{X}_i; \mathcal{W}_t) \end{aligned} \tag{3.1}$$

Least confidence

The probability of the most probable class for an instance is called the confidence. If it is low, the uncertainty of the model for this instance is high. Thus, we can use the negative of the confidence as an uncertainty measure [74].

$$lc_i = -\max_j p_j(y_i|\mathbf{X}_i) \tag{3.2}$$

Least margin

If the probability of the most probable class shows a large difference from the probability of the second-most probable class, the prediction can be regarded as certain. The margin is the difference between the values of the highest posterior probability and the second highest posterior probability [70]. Thus, the negative of the margin is considered as an uncertainty estimator. j_1 and j_2 in the following equation represent the first and second most probable class labels classified by the model.

$$lm_i = -\left(p_{j_1}(y_i|\mathbf{X}_i) - p_{j_2}(y_i|\mathbf{X}_i)\right) \tag{3.3}$$

Entropy

Entropy is considered as an uncertainty estimator which uses all class label probabilities [30].

$$en_i = -\sum_{j=1}^{C} p_j(y_i|\mathbf{X}_i) \log p_j(y_i|\mathbf{X}_i) \tag{3.4}$$

Another approach for estimating the uncertainty is to use a Bayesian model. A Bayesian model is capable of estimating uncertainty, because it produces a distribution of predictions. However, a Bayesian model is not suitable for practical usage, because it is computationally intensive in terms of both training and inference. Thus, we adopt MC dropout [21] technique for Bayesian approximation of a CNN model. Generally, dropout is used in the training process to solve the overfitting problem. If inferences are done with dropout activated, the model produces a different output for each forward pass. The uncertainty for an instance is estimated based on the statistics of the outputs from multiple forward passes of the instance. So, in our system, dropout is activated when the model is being updated or when estimating the uncertainty through MC dropout. Several uncertainty estimation methods based on MC dropout are introduced as below.

Predictive entropy

The entropy is calculated using the average of the results of the multiple forward passes.

$$pe_i = -\sum_{j=1}^{C} \left(\frac{1}{T} \sum_{t=1}^{T} p_j^t(y_i|\mathbf{X}_i) \right) \log \left(\frac{1}{T} \sum_{t=1}^{T} p_j^t(y_i|\mathbf{X}_i) \right) \tag{3.5}$$

Bayesian active learning by disagreement (BALD)

This measure corresponds to the mutual information between predictions and model parameters $\mathcal{W}$ [29]. If a model has high uncertainty regarding an instance on average, but some forward passes make predictions that disagree with each other with high certainty, that instance will maximize this measure. This measure is calculated as follows.

$$bd_i = pe_i - \frac{1}{T} \sum_{t=1}^{T} \left(-\sum_{j=1}^{C} p_j^t(y_i|\mathbf{X}_i) \log p_j^t(y_i|\mathbf{X}_i) \right) \tag{3.6}$$

Variation ratio

In the multiple forward passes, the relative ratio of the number of times classified into each class to the total can be regarded as a probability. Therefore, the largest value of this ratio represents the degree of certainty, and similar to confidence, the smaller the value, the higher the uncertainty [20]. This measure is calculated as follows.

$$vr_i = 1 - \max_i \left(\frac{1}{T} \sum_{t=i}^{T} \mathbb{1} \left(i = \arg\max_j p_j^t(y_i|\mathbf{X}_i) \right) \right) \tag{3.7}$$

Mean standard deviation (mean-STD)

The standard deviation of the probability is calculated for each class, and the average is used [33].

$$ms_i = \frac{1}{C}\sum_{j=1}^{C}\sqrt{\frac{1}{T}\sum_{t=1}^{T}p_j^t(y_i|\mathbf{X}_i)^2 - \left(\frac{1}{T}\sum_{t=1}^{T}p_j^t(y_i|\mathbf{X}_i)\right)^2} \tag{3.8}$$

3.2.4 Query wafer selection

Simple top-K selection

We can calculate the uncertainty with the methods mentioned above for each unlabeled wafer. Assuming that K unlabeled wafers were selected and queried, it is common that K wafers with high uncertainty are selected. This selection is commonly used for uncertainty-based active learning of a CNN [84, 22].

Diversified top-K selection

This situation, which requires selecting K query instances at a time, corresponds to a batch mode active learning [72]. The K wafers selected through the simple top-K selection are likely to have similar spatial patterns, and thus can provide redundant information. Therefore, when sampling the K wafers, the diversity must be considered.

To prevent redundant selection of similar wafers, we propose to use a diversified top-K selection method. Instead of simply choosing the K wafers with the highest uncertainty among all wafers, we select the wafers with the highest uncertainty

within each of the predicted classes.

3.2.5 Query wafer labeling

After the query wafers are chosen, the engineers proceed to label them manually, using visual recognition. These newly-labeled wafers are then merged into the existing labeled-wafers set.

3.2.6 Model update

Once the labeled wafer set has been updated, a new CNN model must be built. However, training the model from scratch takes a long time thus inefficient. To make it efficient, we fine-tune the existing CNN model of the previous phase with the updated labeled set. This method is more time-efficient than training the model from scratch because the model from the previous phase already has information about existing labeled wafers. We set the loss for the newly-labeled wafer larger than the loss for the existing labeled wafer, so that the information in the newly-labeled wafer can be learned quickly while maintaining the information in the existing labeled wafer. The objective function used for fine-tuning can be described as follows. λ is a hyperparameter that represents the ratio of the loss for the newly-labeled wafer to the loss for the existing labeled wafer. It should be larger than 1.

$$\begin{aligned}\min_{\mathcal{W}} \; & -\frac{\lambda}{K} \sum_{\{i|\mathbf{X}_i \in Q\}} \sum_{j=1}^{C} \mathbb{1}\left(y_i = j\right) \log p_j(y_i|\mathbf{X}_i) \\ & -\frac{1}{|\mathcal{D}^L|} \sum_{\{i|\mathbf{X}_i \in \mathcal{D}^L\}} \sum_{j=1}^{C} \mathbb{1}\left(y_i = j\right) \log p_j(y_i|\mathbf{X}_i)\end{aligned} \tag{3.9}$$

3.3 Experiments

3.3.1 Data description

We conducted experiments to demonstrate the effectiveness of the proposed wafer map pattern classification system. The dataset used in the experiment is the WM-811K dataset, which is a real-world fabrication dataset. The dataset[1] was first released in [86] It consists of 811,457 wafer maps from a total of 46,294 lots. The defect type is labeled for approximately 20% of the maps, or 172,950 wafer maps. In this experiment, only the 172,950 labeled instances were utilized. There are nine classes in total: *Non-Pattern*, *Edge-Ring*, *Edge-Loc*, *Center*, *Loc*, *Scratch*, *Random*, *Donut*, and *Near-Full*. Figure 3.2 shows examples of each class.

Table 3.2 shows the distribution of classes in the dataset, which are highly imbalanced. The *Non-Pattern* class occupies an overwhelming proportion, whereas the *Donut* and *Near-Full* classes occupy 0.3% and 0.1%, respectively. As in [86], we used 54,355 wafer maps of the dataset to build a wafer map pattern classification system. The remaining 118,595 wafer maps were used to evaluate the performance of the system. Because CNN requires fixed-size inputs, all wafer maps were resized to (64,64).

3.3.2 Experimental design

In the experiments, we simulated the proposed wafer map pattern classification system. Similar to the related literature [57, 44], we used a LeNet-5 [45]-like CNN architecture shown in Figure 3.3. Our architecture is relatively smaller than modern CNN architectures such as AlexNet [43] and VGGnet [77]. This is adequate for our

[1] http://mirlab.org/dataset/public/

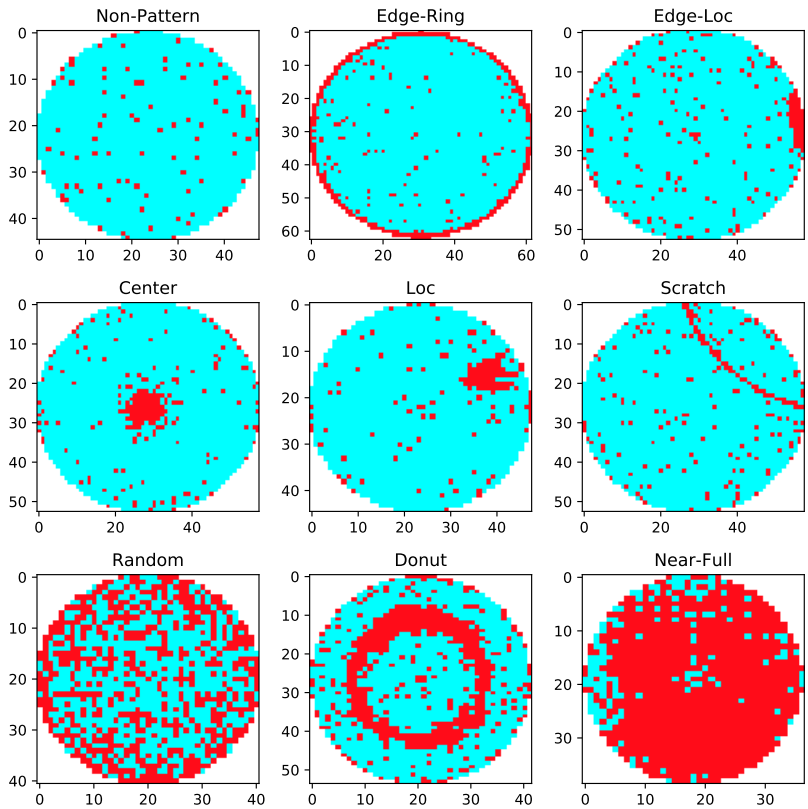


Figure 3.2: Examples of wafer maps and their labels.

Table 3.2: Description for dataset used in Chapter 3.

Defect Type	No. Train	No. Test	No. Total
Non-Pattern	36,730	110,701	147,431
Edge-Ring	8,554	1,126	9,680
Edge-Loc	2,417	2,772	5,189
Center	3,462	832	4,294
Loc	1,620	1,973	3,593
Scratch	500	693	1,193
Random	609	257	866
Donut	409	146	555
Near-Full	54	95	149
Total	54,355	118,595	172,950

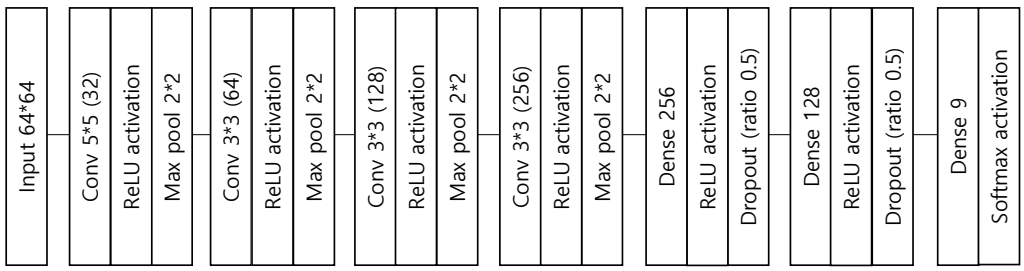


Figure 3.3: CNN architecture used for the experiments.

problem because wafer maps are much simpler than conventional image representations and the prediction model must be trained with a small amount of instances in the active learning setting. At the beginning, 400 randomly selected wafers were labeled. Setting aside 200 wafers as the validation set, the other 200 wafers were used as the initial training set. All the remaining wafers were used as the initial unlabeled set. This corresponds to real-world situations, where engineers inspect randomly selected unlabeled wafer maps. Extreme minority classes, such as *Donut* and *Near-Full*, may not have been included in the initial training set. For the query wafer selection step, the query size K was set to the same as the number of classes. The query wafer labeling step was simulated by revealing the corresponding labels in the original dataset.

We used the Adam optimizer [41] with a learning rate of 0.001 and a batch size of 128. Each model training was terminated if the validation loss failed to decrease over 20 consecutive phases. For the model update step, the hyperparameter λ in Equation (3.9) was set to 10 by conducting a preliminary experiment to investigate its effect on the performance. For the uncertainty estimation step, the number of forward passes was set to 50 when MC dropout was used.

We conducted experiments on the seven uncertainty estimation methods described in Section 3.2.3, and applied the two wafer selection methods described in Section 3.2.4, for a total of 14 combinations. To succinctly represent each combination, suffixes '_d' or '_s' are used to indicate diversified top-K selection and simple top-K selection, respectively. As baselines, we used BatchBALD [42], random selection method, and full model. The random selection method randomly selects wafers to be labeled without uncertainty estimation. The full model is trained with the entire training set, assuming that the dataset is completely labeled.

The performance of the CNN model for wafer map pattern classification was evaluated on the test set in terms of the area under the receiver operating characteristic curve (AUROC). Dropout in the model is deactivated when making predictions for performance evaluation. Because the original AUROC is for binary classification, we used a multi-class modification of the AUROC [65]. The multi-class AUROC is calculated by taking the average of multiple AUROCs, each of which having been obtained separately by binary classification of discriminating a single class from all of the other classes.

All experiments were performed with 10 independent repetitions with different random seeds. We report an average over the 10 repetitions.

3.3.3 Results and discussion

Figure 3.4 shows the comparison results of the seven uncertainty estimation methods with the diversified top-K selection. The results of the three baseline methods are also shown. The X-axis represents the repetition phase of the proposed active learning system. The performance increased as the phase progressed. As shown in the figure, all seven of the uncertainty estimation methods were superior to the random

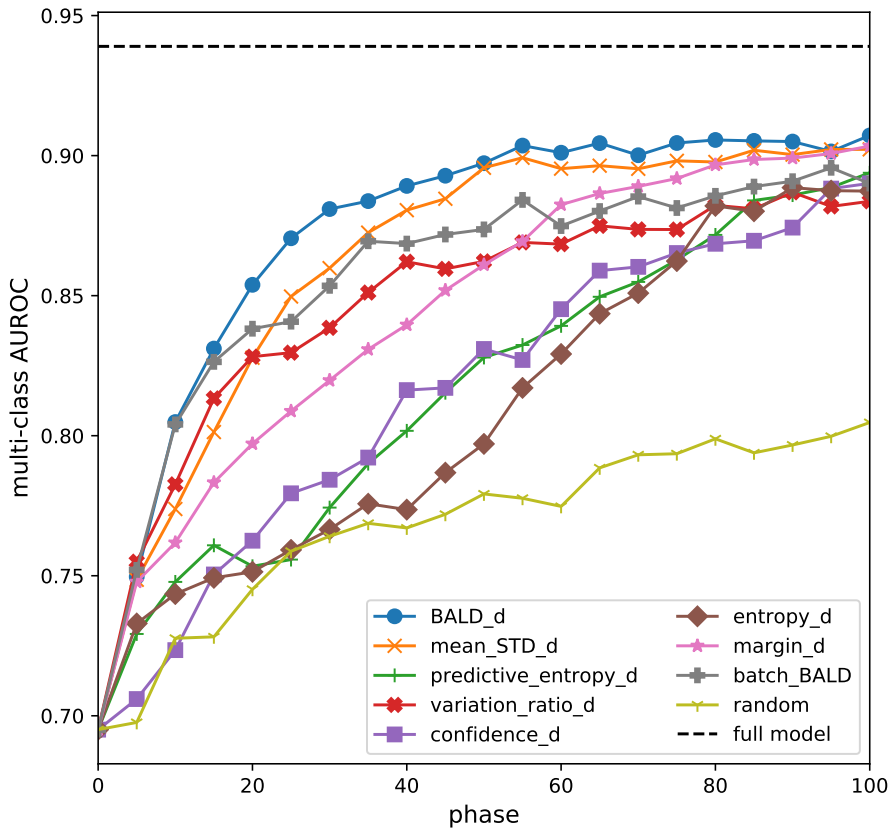


Figure 3.4: Overall comparison of uncertainty estimation methods with diversified top-K selection.

selection method. While the random selection method showed a slower performance increase per phase, proposed method had a faster performance increase. Among the seven uncertainty estimation methods, BALD and mean-STD showed the best performances, in order. The performance of these two methods at the 60th phase was quite close to that of the full model. The BatchBALD was not superior to the other methods, as it has been known to perform worse under class imbalance [42].

We assessed the overall performance across 9 defect type classes and 10 different initial training sets using Dolan-More curve [17]. For the 90 problems, $\rho(\tau)$ indicates

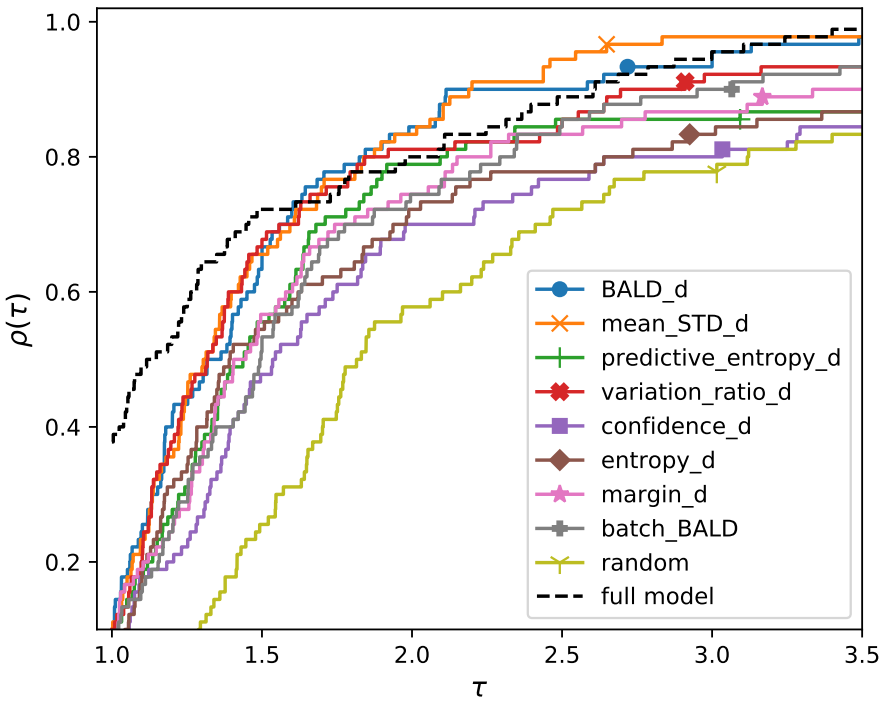


Figure 3.5: Dolan-More curves of the CNN models at the 60th phase.

the fraction of the problems that the method's error rate is not greater than τ times the best error rate. The Dolan-More curves of CNN models at the 60th phase for the compared methods are presented in Figure 3.5. The full model yielded the highest $\rho(\tau)$ at τ=1, as the fraction of problems with the lowest error rate was the highest. With increased τ, $\rho(\tau)$ of BALD and mean-STD became greater than that of the other methods including the full model, meaning that BALD and mean-STD evenly performed well across the problems.

The comparison results for each of the defect types are shown in Figure 3.6. In active learning for a class imbalance problem, it is important to select instances of the minority class. In the case of *Near-Full* which has the least instances, the

performance difference between the uncertainty estimation methods was particularly severe. The performances differed greatly depending on whether or not the instances corresponding to the defect types were included enough in the training set. BALD and mean-STD outperformed other methods. For other minority classes, *Random* and *Donut*, the proposed system showed high performance as well.

For the *Non-Pattern*, *Edge-Loc*, *Loc*, and *Scratch*, the performance of the proposed method was far worse than the full model and increased slowly by active learning. These classes have relatively large variations in their spatial defect patterns. So, they require a large amount of various training instances. Performance differences between the uncertainty estimation methods were also significant for these classes. This suggests that it is more important to select informative training wafer maps. Furthermore, the proposed method even outperformed the full model for some particular classes, which demonstrates that using a small informative training set can be better than using the whole dataset.

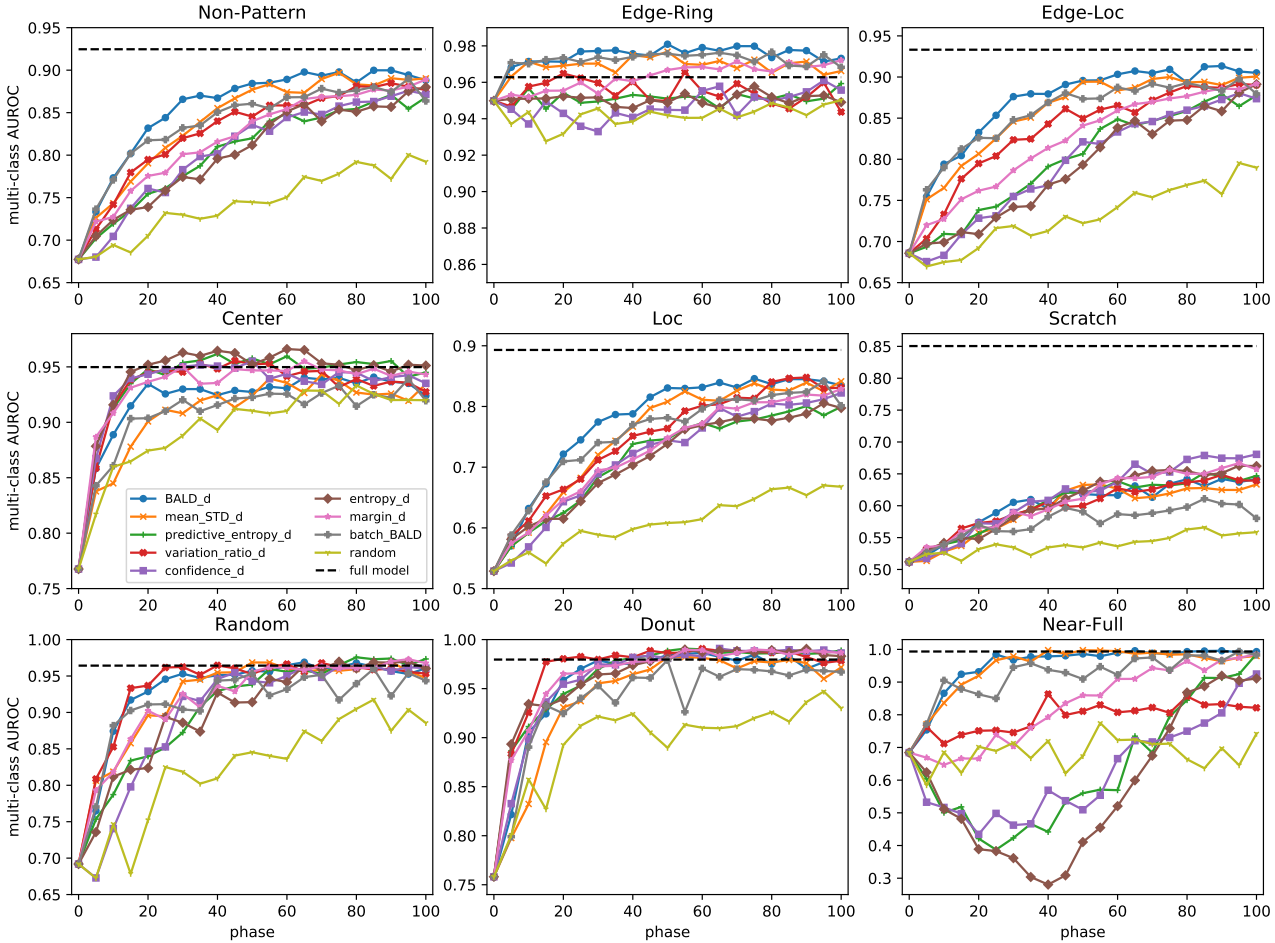


Figure 3.6: Comparison of uncertainty estimation methods with diversified top-K selection for each defect type. Each graph has a different y-axis range to clarify the performance difference between methods.

Figure 3.7 compares the performance between the diversified top-K selection and simple top-K selection. In most uncertainty estimation methods, the diversified top-K selection yielded performance similar to or better than the simple top-K selection, which indicates that the diversified top-K selection helps to choose more informative wafer maps to improve the classification performance. Above all, the performance of the diversified top-K selection was superior when used with BALD, which showed the best performance.

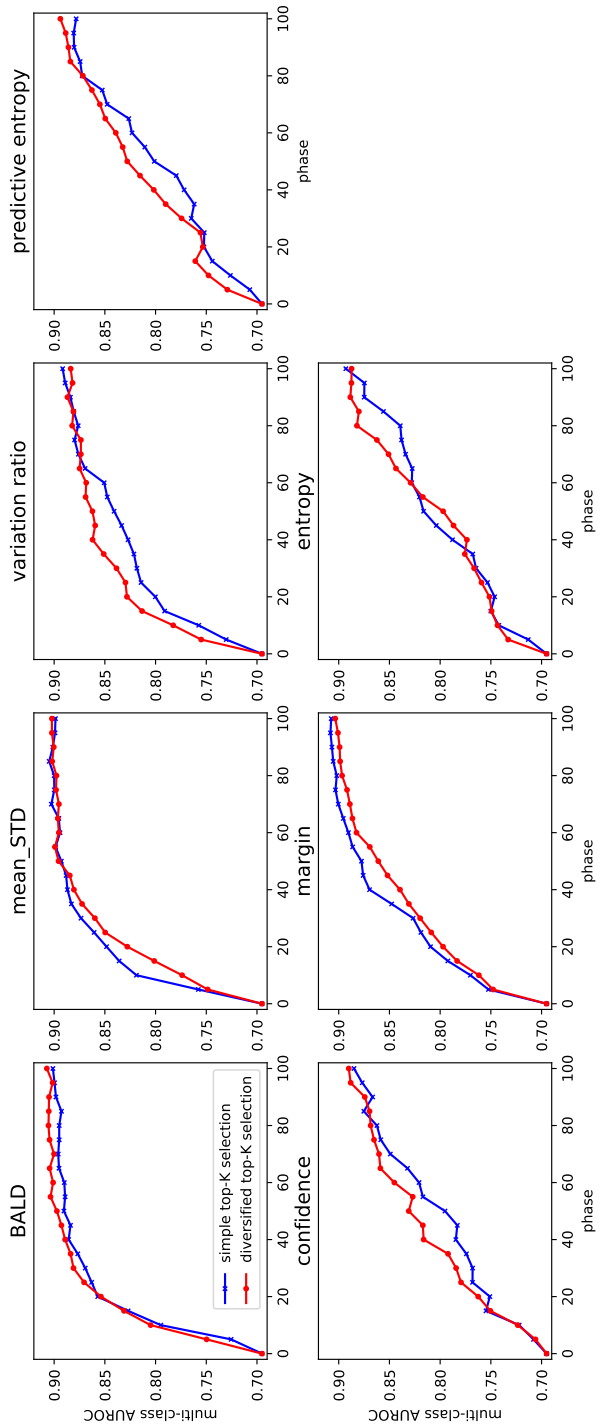


Figure 3.7: Comparison between the diversified top-K selection and simple top-K selection.

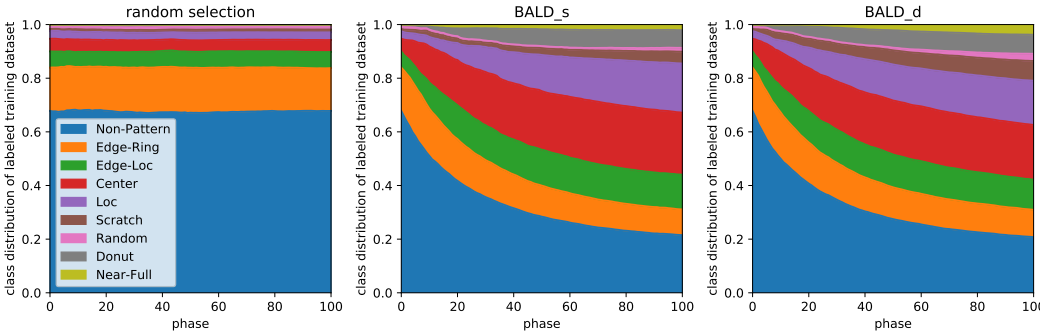


Figure 3.8: Class distributions of the labeled training set as the phase progressed.

To investigate how the proposed method works, we visualized in Figure 3.8 the change in the class distribution of the labeled training set with the phase progress for the random selection method, BALD_s, and BALD_d. The random selection method showed no change in class distribution, whereas BALD selected minority class instances more so that the proportion of minority classes increased with the progress. This demonstrate that the proposed method selects informative instances without duplication. Additionally, BALD_d exhibited more balanced class distribution than BALD_s. This suggests that diversified top-K selection effectively alleviate the class imbalance problem.

Chapter 4

Active Cluster Annotation for Wafer Map Pattern Classification

4.1 Problem Description

This chapter also solves the label cost problem of WMPC as in Chapter 3. The automation of WMPC has been studied extensively [16, 10]. Recently, CNN-based WMPC has been actively and successfully investigated. To construct a high-performance classification model using a CNN, a large amount of labeled training data is required. However, it is expensive to secure a large amount of labeled wafer maps, as the wafer maps must be annotated by expert engineers. In Chpater 3, we solved this problem using an active learning strategy [76]. By selectively annotating informative wafer maps based on the classification uncertainty, the performance of the WMPC was improved cost-effectively.

In the conventional active learning framework, a single wafer map is annotated in one interaction by an engineer. However, this wafer map-level annotation is inefficient because there exist wafer maps with similar patterns in practice. If we can constitute a cluster of wafer maps having similar defect patterns, then the engineer can conduct cluster-level annotation, in which multiple wafer maps are annotated in one interaction. This cluster-level annotation does not differ significantly from wafer map-level annotation in terms of engineer's annotation cost. Therefore, in this work,

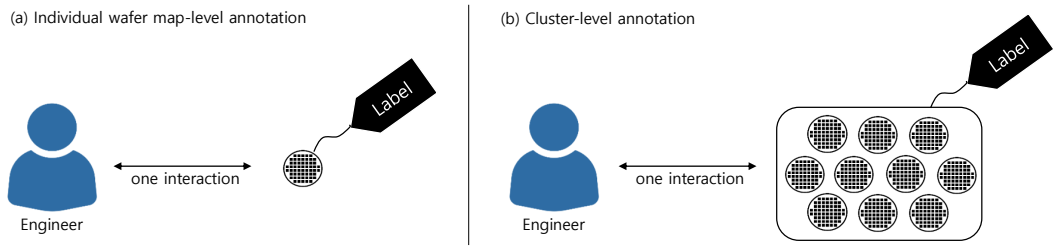


Figure 4.1: Illustrative explanation of cluster-level annotation. In individual wafer map-level annotation, one wafer map is annotated in one interaction; in cluster-level annotation, multiple wafer maps are annotated in one interaction.

we assume that the costs of the wafer map-level and cluster-level annotations are identical. The cluster-level annotation is depicted in Figure 4.1.

Based on this idea, we propose an active cluster annotation for WMPC to achieve a better performance with reduced annotation cost. For a dataset annotated only for a small subset of wafer maps, clustering is first conducted on unlabeled wafer maps. When an active learning iteration begins, a CNN is constructed with labeled wafer maps. Subsequently, cluster-level classification uncertainties for unlabeled wafer maps are evaluated using the CNN. The clusters to be annotated are selected based on the uncertainties and then assigned to the engineer. The selected clusters are annotated at the cluster level and merged into the set of labeled wafer maps. The next iteration begins with an increase in the number of labeled wafer maps. By repeating the iterations, a high-performance CNN can be achieved with reduced annotation cost. The effectiveness of the proposed method is demonstrated through experiments using real-world data collected from a semiconductor manufacturer.

4.2 Proposed Method

In this section, we introduce the structure of the proposed active cluster annotation. The purpose of the framework is to build a CNN that classifies wafer maps into defect types with high accuracy while reducing manual annotation costs by engineers. The key difference from the existing active learning framework is that the annotation is conducted at the cluster level.

Figure 4.2 shows the flow chart of the active cluster annotation framework, which comprises five steps: (1) clustering of unlabeled data, (2) CNN training with labeled data, (3) cluster-level uncertainty estimation, (4) query cluster selection, and (5) cluster-level annotation. After step (1) is conducted, steps (2) to (5) are repeated.

Table 4.1 shows the notations used to describe the proposed framework, and Algorithm 2 presents the pseudocode of the entire process. First, clustering is conducted on unlabeled wafer maps $\mathcal{D}^U$ to obtain K clusters, $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$. When active learning iteration begins, the CNN is first constructed with a small amount of initial labeled wafer maps $\mathcal{D}^L$. Subsequently, classification uncertainties of the CNN for unlabeled wafer maps are estimated, and cluster-level uncertainties are calculated for each $\mathcal{C}_i$. With these cluster-level uncertainties, q clusters to be annotated are selected and delivered to the engineer. After the engineer annotates the clusters, all the wafers in the clusters are merged into labeled wafer maps $\mathcal{D}^L$. The next iteration begins as the CNN is trained with an increased amount of labeled wafer maps. As the iterations are repeated, the performance of the CNN improves gradually. The following subsections describe each step in detail.

The proposed method can be explained in comparison with self-training, a method for semi-supervised learning. The self-training method also contains the process of

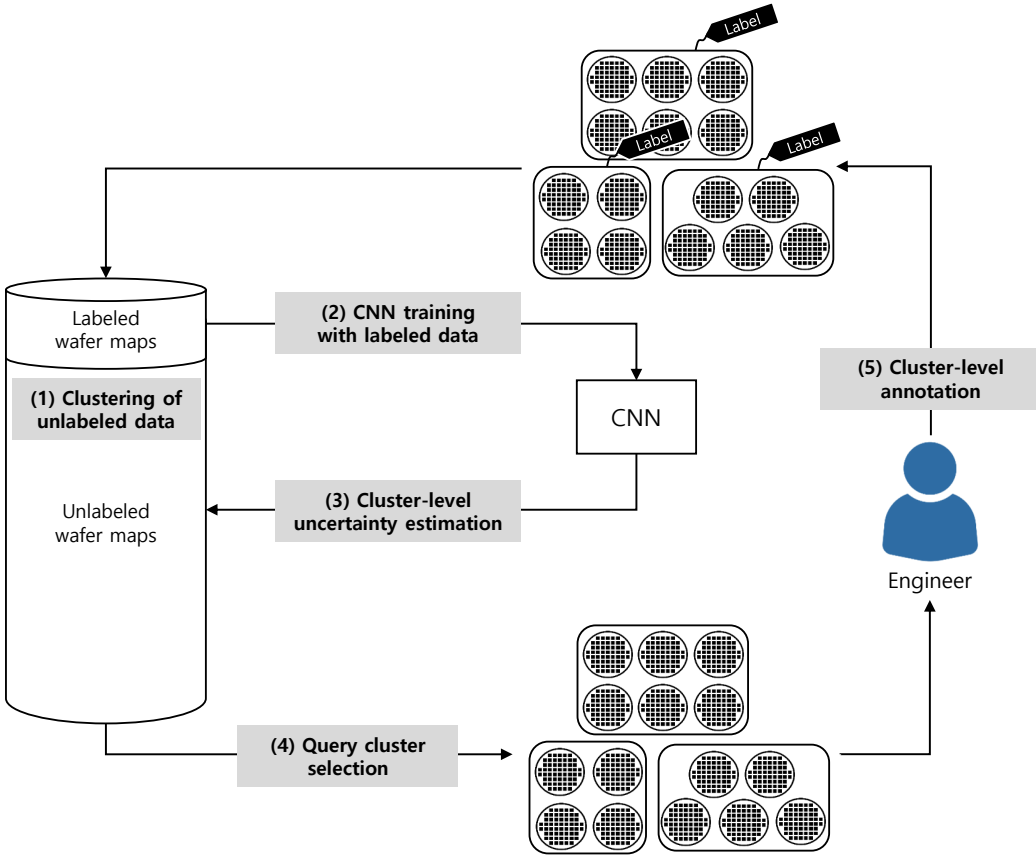


Figure 4.2: Flow chart of active cluster annotation framework.

Table 4.1: Notations used in Chapter 4.

Notation	Type	Description
$\mathcal{D}$	set	whole wafer map dataset.
$\mathcal{D}^L$	set	labeled subset of $\mathcal{D}$.
$\mathcal{D}^U$	set	unlabeled subset of $\mathcal{D}$
$\mathbf{X}_i$	matrix	i-th wafer map in $\mathcal{D}$
y_i	scalar	label for $\mathbf{X}_i$
d	scalar	number of defect types
q	scalar	number of clusters to be queried for each iteration
$\mathcal{C}_k$	set	k-th cluster of wafer maps in $\mathcal{D}^U$
K	scalar	total number of clusters in $\mathcal{D}^U$ at the beginning.
$\lvert\mathcal{C}\rvert_{average}$	scalar	average number of wafer maps per cluster
$\mathcal{W}$	set	CNN parameters

Algorithm 2 Pseudocode of proposed framework

Input: Labeled dataset $\mathcal{D}^L$, Unlabeled dataset $\mathcal{D}^U$
Output: CNN parameters $\mathcal{W}$

procedure
  $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K \leftarrow$ Clustering wafers in $\mathcal{D}^U$
  $\mathcal{W} \leftarrow$ Construct CNN with $\mathcal{D}^L$
  while not reach maximum iteration **do**
    Estimate uncertainty for each wafer map in $\mathcal{D}^U$
    Calculate representative uncertainty for each cluster $\mathcal{C}_k$
    $Q \leftarrow q$ most informative clusters from $\mathcal{D}^U$
    Annotate each cluster in Q
    $\mathcal{D}^L \leftarrow \mathcal{D}^L \cup Q$, $\mathcal{D}^U \leftarrow \mathcal{D}^U \setminus Q$
    $\mathcal{W} \leftarrow$ Train CNN with $\mathcal{D}^L$
  end while
  return $\mathcal{W}$
end procedure

performing prediction on unlabeled data. The predicted class for the instance with low classification uncertainty is utilized as a pseudo label, and the pseudo labeled instances are merged into labeled data. Then, the model is reconstructed with the increased amount of labeled data. Unlike self-training, in the active cluster annotation, clusters with high classification uncertainty is rather selected and labeled by a labeler rather than with the predicted class. It can be seen that all instances in the cluster are pseudo-labeled with the same class by the labeler.

4.2.1 Clustering of unlabeled data

In the proposed framework, uncertainty estimation, query selection, and annotation are conducted at cluster level rather than wafer map level. In this regard, clustering is conducted in this step such that all the wafer maps in $\mathcal{D}^U$ belong to each cluster, $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K$. This clustering is extremely important in the proposed framework because cluster-level annotation can generate label noise if the wafer maps in a

cluster are inconsistent.

To obtain the desired clustering result, the raw wafer map is not suitable as the input for clustering. If the raw wafer maps are used in their original form, wafers belonging to the same defect type but having slightly different locations of defective dies can be assigned to completely different clusters. Therefore, to obtain the desired clustering result, manually extracted features that can represent the characteristics of the wafer maps can be used as the input for clustering.

In this study, the following density-based, radon-based, and geometry-based features are extracted and constitute a vector to represent a wafer map. The constructed vectors are used as feature vectors for clustering. They are conventional features frequently used in WMPC research. With a total of 59 extracted features, the k-means clustering algorithm is used in this study.

- Density-based feature [18, 69] : The wafer map is divided into 13 zones and the density of defective dies in each zone is calculated. The defect density values for these 13 zones are used as features.

- Radon-based feature [86, 64, 69] : Radon transform is perfomred to generate multiple vector representations by projecting a wafer map at various angles. The average and standard deviation of the multiple vectors are calculated. Subsequently, cubic interpolation is applied such that these vectors have 20 lengths each. A total of 40 elements of these two vectors are used as features.

- Geometry-based feature [86, 18, 69] : To obtain the most salient region for the defect type, noise filtering is performed. Six geometric features corresponding to the perimeter, area, length of minor axes, length of major axes, defect ratio,

and eccentricity are extracted for the salient region.

4.2.2 CNN training with labeled data

A CNN is a representative deep learning methodology. It can use raw images as input and extract suitable features from them implicitly. Therefore, there is no need for a domain expert to extract features manually. CNNs are generally composed of convolution layers, pooling layers, and fully-connected layers. The convolution layers extract features from the input, whereas the pooling layers reduce the dimension of the features. Finally, classification is performed using fully-connected layers. A CNN exhibits shift/rotation invariance because the filters in the convolution layers are shared across the entire image and down-sampling is conducted in the pooling layer [24]. These properties of the CNNs facilitate WMPC.

In this step, the CNN is trained with labeled wafer maps using two different methods. One is to train from scratch with randomly initialized parameters in every iteration and the other is to train incrementally using the existing CNN of the previous iteration as in Chapter 3. Incremental learning can reduce the training time. When training the model incrementally, the objective function can be described as follows: To render the loss of the newly-labeled wafer map greater than that of the existing labeled wafer map, the hyperparameter λ that represents the ratio between two types of loss should be greater than one.

$$\begin{aligned}\min_{\mathcal{W}} \; & -\frac{\lambda}{|Q|} \sum_{\{i|\mathbf{X}_i \in Q\}} \sum_{j=1}^{d} \mathbb{1}\left(y_i = j\right) \log p(y_i = j|\mathbf{X}_i) \\ & -\frac{1}{|\mathcal{D}^L|} \sum_{\{i|\mathbf{X}_i \in \mathcal{D}^L\}} \sum_{j=1}^{d} \mathbb{1}\left(y_i = j\right) \log p(y_i = j|\mathbf{X}_i)\end{aligned} \tag{4.1}$$

4.2.3 Cluster-level uncertainty estimation

To select and annotate at cluster level rather than at the wafer map level, uncertainties per cluster rather than per wafer map are required. First, the classification uncertainties of unlabeled wafer maps are calculated using the current CNN. Subsequently, for the cluster-level uncertainty, the representative uncertainty for a cluster can be calculated by using a summary statistic, such as the maximum, minimum, and average.

To calculate the classification uncertainties of the CNN for unlabeled wafer maps, the least margin method is used because in the previous chapter, it showed the best performance among the simple uncertainty sampling approaches which uses the output of the last softmax layer. This method defines the difference between the probabilities of the most probable and second most probable classes as the margin. The negative of the margin is considered as an uncertainty value u, as in the following equation. j_1 and j_2 represent the most and second most probable classes, respectively.

$$u_i = -\left(p(y_i = j_1|\mathbf{X}_i; \mathcal{W}) - p(y_i = j_2|\mathbf{X}_i; \mathcal{W})\right) \tag{4.2}$$

As a summary statistic for cluster-level uncertainty, the minimum value is used in this study. Then, the representative uncertainty for a cluster $\mathcal{C}_k$ can be expressed as $\min_{\{i|\mathbf{X}_i \in \mathcal{C}_k\}} u_i$. The final performance did not differ significantly when the minimum, maximum, and average values were used.

4.2.4 Query cluster selection

In this step, q clusters to be queried are selected based on the uncertainty for each cluster. In general, q clusters with the highest uncertainty are selected as informative. However, these q clusters are likely to be similar and provide duplicate information. Hence, the diversified top-q selection method proposed in the previous chapter is utilized in this study (referred to as diversified top-K selection in the previous chapter).

In the original diversified top-q selection, the selected instances are diversified with the predicted classes. To apply the diversified top-q selection at the cluster level, cluster-level predicted classes are required. In this study, a mode of predicted classes of wafer maps are used as the cluster-level predicted classes. In other words, clusters with the highest uncertainty within each cluster-level predicted class are selected and queried.

4.2.5 Cluster-level annotation

The selected q clusters are provided to the engineer. The engineer visually inspects and annotates each cluster. At this time, inconsistent wafers may be included in each cluster, but the cluster is annotated as the majority of defect types of wafers belonging to the cluster. When a cluster is annotated, all the wafers belonging to the cluster are equally annotated. Subsequently, the annotated wafers are merged into the labeled training set. This cluster-level annotation can secure more labeled wafer maps with the same number of annotation tasks compared with individual wafer map-level annotation.

4.3 Experiments

4.3.1 Data description

We conducted experiments to demonstrate the effectiveness of the proposed method. The dataset used in the experiment is the WM-811K dataset, which is a real-world fabrication dataset. This dataset was first released in [86]. It comprises 811,457 wafer maps, of which 172,950 wafer maps are labeled. In this experiment, 172,946 labeled wafer maps were used, excluding four wafers whose features to be used for clustering could not be defined because the number of dies was less than 200. As shown in Figure 3.2, nine defect types were observed: *Non-Pattern*, *Edge-Ring*, *Edge-Loc*, *Center*, *Loc*, *Scratch*, *Random*, *Donut*, and *Near-Full*.

Table 3.2 shows the distribution of defect classes. It is a highly imbalanced dataset, in which *Non-Pattern* constitutes a large portion and *Near-Full* a small portion. The training and test sets were divided in the same way as in [86]. The training set was used to simulate the proposed framework and the test set was used to evaluate the performance of the CNN in the framework. All wafer maps were resized to (64,64).

4.3.2 Experimental setting

The proposed framework was simulated experimentally. Among the 54,354 wafer maps in the training set, 400 randomly selected wafers were treated as the initial labeled set. Among them, 200 wafers were used for training the initial CNN, and the other 200 wafers were used as the validation set. The remaining 53,954 wafer maps were used as the initial unlabeled set. The annotation process by the engineer was simulated by using the original labels of the wafer maps. In cluster-level annotation,

Table 4.2: Description for dataset used in Chapter 4.

Defect Type	No. Train	No. Test	No. Total
Non-Pattern	36,729	110,698	147,427
Edge-Ring	8,554	1,126	9,680
Edge-Loc	2,417	2,772	5,189
Center	3,462	832	4,294
Loc	1,620	1,973	3,593
Scratch	500	693	1,193
Random	609	257	866
Donut	409	146	555
Near-Full	54	95	149
Total	54,354	118,592	172,946

all the wafers in the cluster were annotated as the most frequent class in their original labels. If the cluster-level annotation is simulated through other representative values, the result may be slightly different, but annotating with the most frequent class could be the most similar to the actual situation.

The architecture of the CNN used is same as the architecture used in the previous chapter. An Adam optimizer with a learning rate of 0.001 was used for model training, and the batch size was set to 128. Early stopping was applied if the validation loss did not decrease over 20 consecutive epochs. When using incremental learning to update the CNN in each iteration, the hyperparameter λ in the objective function in (4.1) was set to 10. For the cluster selection step, the query size q for each iteration was set to the the number of classes. All the experimental settings mentioned above are the same as those used in the previous chapter.

To investigate the effect of the cluster size, the average number of wafers in each cluster, $|\mathcal{C}|_{average}$, was varied to 10, 30, 50, 70, and 90. It was controlled through the hyperparameter K of the k-means clustering algorithm, *i.e.*, the K values were set to 5395, 1798, 1079, 770, and 599 by $K = \left\lfloor \frac{|\mathcal{D}^U|_{begin}}{|\mathcal{C}|_{average}} \right\rfloor$. As a baseline, we used the

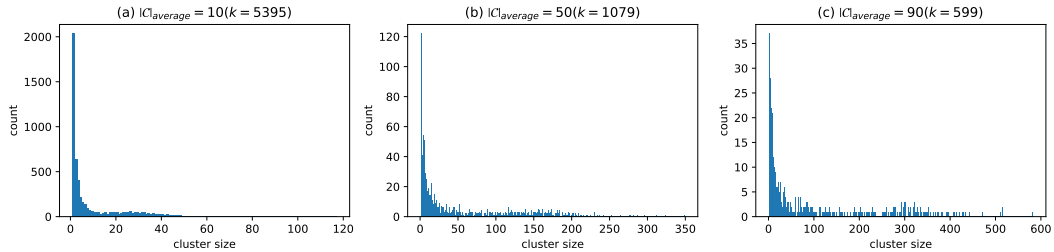


Figure 4.3: Histogram of cluster size according to $|\mathcal{C}|_{average}$.

individual wafer map-level annotation method, which was introduced in the previous chapter; this approach is the same as setting the number of cluster K to $|\mathcal{D}^U|_{begin}$. In addition, a random selection method, which randomly selects the clusters to be annotated, and a full model, which is trained with the entire labeled training set, were compared.

The performance of the CNN was evaluated on the test set in terms of the multi-class area under the receiver operating characteristic curve (AUROC). All experiments were performed with 10 independent repetitions and their average was used for the result.

4.3.3 Clustering results

In this subsection, we discuss the clustering results for the unlabeled wafer maps. In Figure 4.3, histograms of cluster size are presented. These distributions have a long tail distribution in which most of the clusters are small, and a few large clusters exist. This is consistent with the fact that an imbalance exists in the defect type.

For the cluster-level annotation to be effective, high-quality clustering is necessary because all the wafers in each cluster are equally annotated. Purity [71] was introduced to evaluate the clustering results. Purity is a measure that indicates

whether each cluster comprises a single class. Assuming that every element in a cluster is assigned as the most frequent class in the cluster, the accuracy of the assignment is purity. Therefore, purity can be regarded as the annotation accuracy when cluster-level annotation is used. Using N as the total number of instances, $\mathcal{C}_i$ the ith cluster, and D_j the set of instances of the jth class, the purity can be expressed by the following equation:

$$purity = \frac{1}{N} \sum_{i=1}^{m} \max_{j} |\mathcal{C}_i \cap D_j| \tag{4.3}$$

Figure 4.4 shows a graph of purity *vs.* $|\mathcal{C}|_{average}$. As $|\mathcal{C}|_{average}$ was increased, purity decreased. Because the purity values exceeded 0.95 for all $|\mathcal{C}|_{average}$, the wafers in each cluster can be regarded as sufficiently consistent. These results indicate that cluster-level annotation is not risky. In addition, since the consistency of wafer maps in each cluster affects the ease with which an engineer annotates a cluster at once, the result supports our assumption that the costs of cluster- and wafer map-level annotations are the same.

4.3.4 Classification performance

With the active cluster annotation, the amount of labeled wafers acquired in one active learning iteration increases. Therefore, we can secure more labeled wafer maps with the same number of annotation tasks compared with individual wafer map-level annotation. Figure 4.5 shows the change in the amount of the labeled wafer map as iteration progresses. As $|\mathcal{C}|_{average}$ value was increased, the amount of labeled wafer maps tended to increase more rapidly as expected.

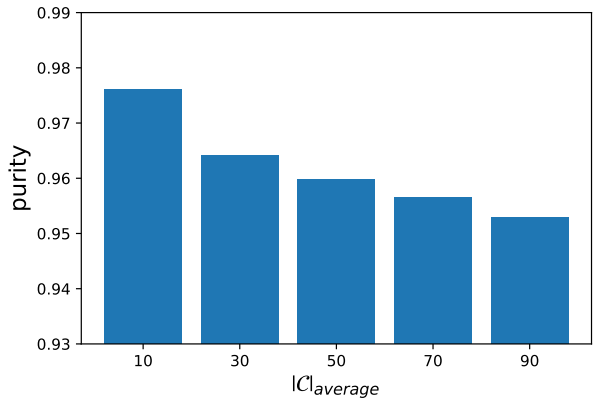


Figure 4.4: Purity according to $|\mathcal{C}|_{average}$.

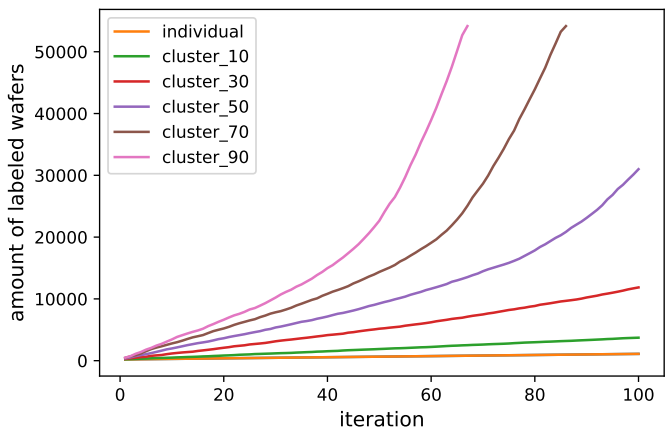


Figure 4.5: The amount of labeled wafer maps as iteration progresses.

Figure 4.6 shows the experimental results of active cluster annotation for different $|\mathcal{C}|_{average}$ and baselines. The X-axis represents the iteration of the active learning framework, whereas the Y-axis represents the AUROC for the CNN on the test set. Each line graph represents each method to be compared. As the iteration progresses, the performance increases; the faster the performance increases, the more cost-effective the method is. When $|\mathcal{C}|_{average}$ is 70 or higher, all unlabeled wafers are annotated before reaching 100 iterations.

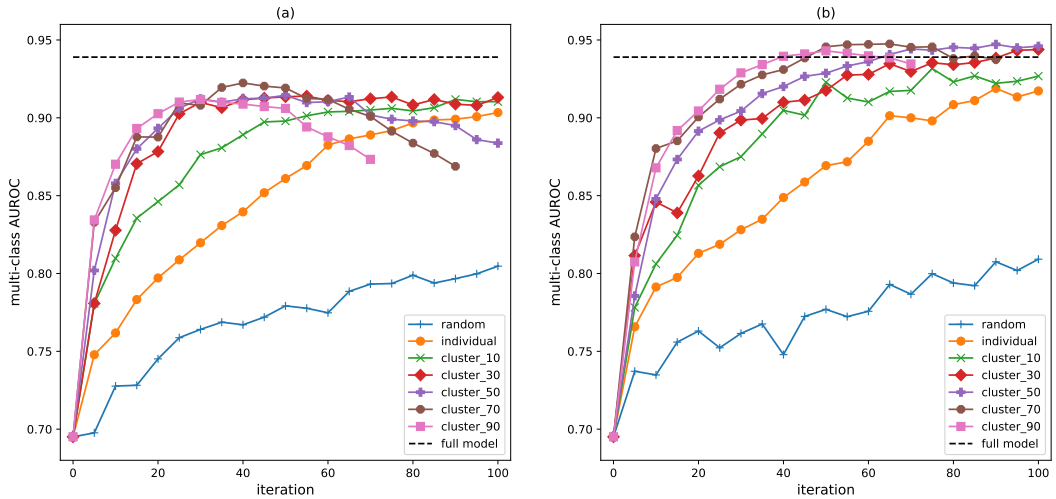


Figure 4.6: Classification performance comparison between methods. (a) incremental learning in each iteration. (b) learning from scratch in each iteration.

Figure 4.6(a) shows the results when the CNN was updated incrementally. Similar to the previous chapter, the individual wafer map-level annotation method based on uncertainty demonstrated better performances than the random selection method. In the early iterations, the performance of the cluster-level annotation method improved at a faster rate compared with the individual annotation method. However, the performance deteriorated as the iteration progressed. Owing to cluster-level annotation, the number of newly-labeled wafer maps per iteration increased and changed dynamically. Therefore, it is conjectured that the overfitting on the newly-labeled wafers occurred.

Figure 4.6(b) shows the results obtained when the CNN was trained from scratch in each iteration. The effect of the incremental learning method is excluded, and only the effect of cluster-level annotation can be verified. As shown in the figure, the proposed cluster-level annotation was superior to the existing individual wafer map-

level annotation. The cluster-level annotation with larger $|\mathcal{C}|_{average}$ showed a faster performance increase. If $|\mathcal{C}|_{average}$ is large, the amount of labeled training data will increase faster, but the label noise due to cluster-level annotation may increase. As shown by the experimental results, the performance increase by the former exerted more impact than the performance degradation by the latter. In particular, when $|\mathcal{C}|_{average}$ was 30, 50, 70, and 90, the active cluster annotation exceeded the full model at the 85th, 60th, 45th, and 40th iterations, respectively.

4.3.5 Analysis for label noise

Figure 4.7 shows the label error rate of the labeled training set as the iteration progresses. The original label of the dataset used in the experiment and the label annotated with cluster-level annotation were compared and considered as an error if they did not match. As shown in the graph, the label error rate increased slightly and then decreased. This is because the clusters with high uncertainty are first selected. A high classification uncertainty means that the pattern is ambiguous. Such clusters are likely to have low purity and hence high label noise. Therefore, the label error rate increased in the beginning and then decreased as clusters with low uncertainties became annotated. When all unlabeled wafers were annotated, the label error rate was equal to $1 - purity$.

Additionally, we visually verified the label noise that occurred in cluster-level annotation. Figure 4.8 shows examples of wafer maps belonging to each cluster and their original labels assigned in the dataset. Figure 4.8(a) shows a cluster annotated as "*Center*" via the proposed method. In this cluster, wafers whose original labels were "*Loc*" or "*Donut*," *i.e.*, not "*Center*," were discovered. These were label noises. However, it was observed that the wafer maps in fact showed a similar pattern to

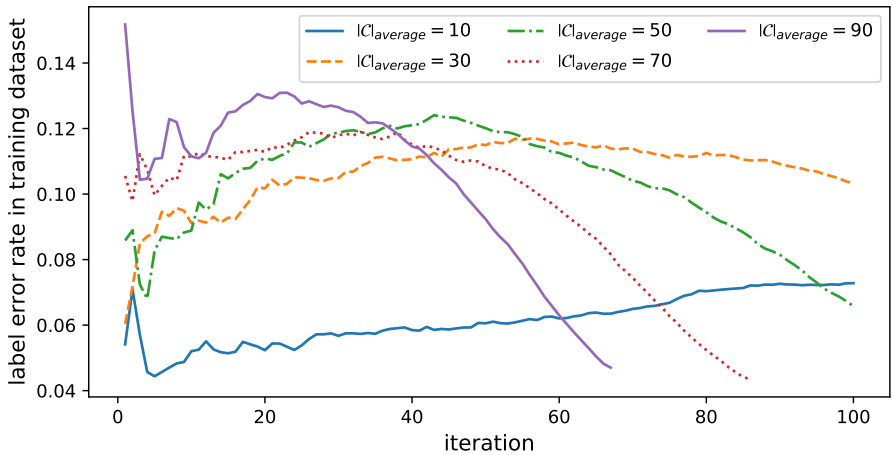
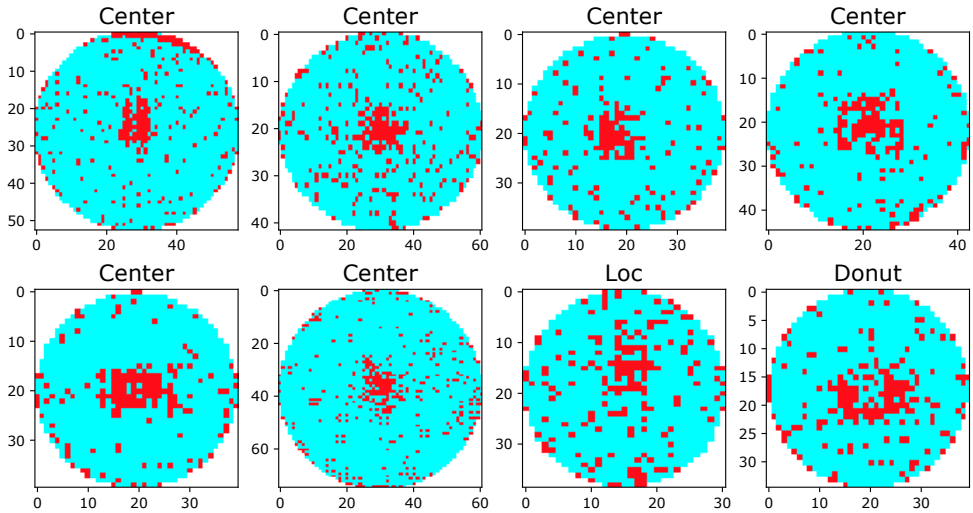


Figure 4.7: Label error rate of labeled training dataset as iteration progresses.

other wafer maps in the cluster. Similarly, Figure 4.8(b) shows a cluster annotated as "*Edge-Loc*" via the proposed method. Wafer maps whose original labels are "*Loc*" but their actual pattern was almost similar to those of other wafer maps in the cluster were discovered. As such, the original labels assigned in the dataset were ambiguous. Rather, it appears that consistent annotation can be achieved through cluster-level annotation. As presented in subsection 4.3.4, the cluster-level annotation method showed better performances than the full model. It can be conjectured that the robustness of the cluster-level annotation method contributed to this result.

(a) A cluster annotated as "*Center*" by proposed method



(b) A cluster annotated as "*Edge-Loc*" by proposed method

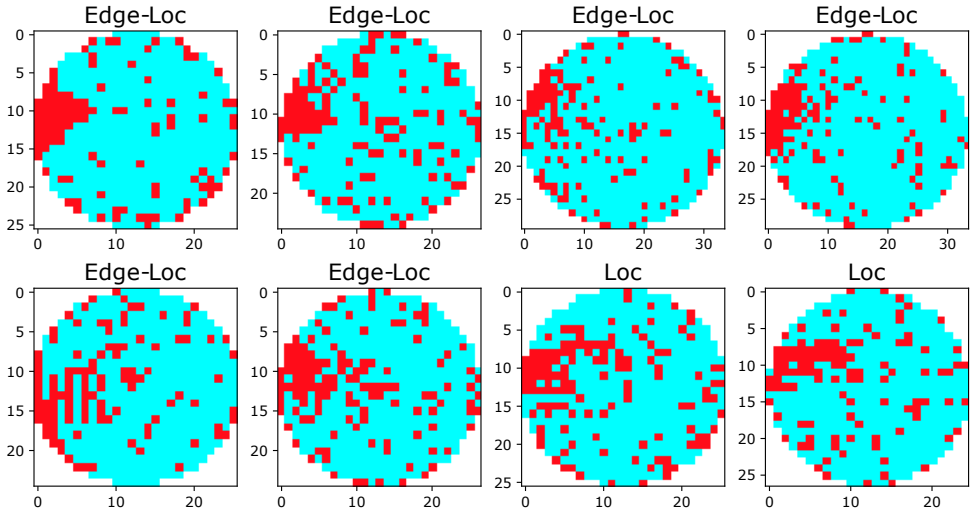


Figure 4.8: Examples of wafer map clusters and their original labels.

Chapter 5

Active Inspection for Fault Prediction

5.1 Problem Description

A manufacturing process typically consists of hundreds of processing steps. To ensure high-quality of manufactured products, intermediate inspections are performed after each step. Products that do not pass predefined specifications are detected to be discarded or fixed. However, not every product can go through all the inspections because of high inspection time and costs. To maintain high productivity of the manufacturing process, manufacturers generally categorize the inspections into basic and advanced inspections. Basic inspections are essential for quality control so all products are subject to. On the other hand, advanced inspections are more strict but expensive so only some sampled products are subject to.

Despite many inspection processes, some products are eventually identified as faulty at the end of the process. To address the issue, several studies attempted to construct a prediction model that detects the faulty products early using inspection results based on machine learning [90, 39, 36, 40]. The products predicted as faulty can be fixed or discarded in advance, which contributes to improving the final yield. The problem situation of predicting the faults through basic inspections and advanced inspections is illustrated in Figure 5.1. Since advanced inspections

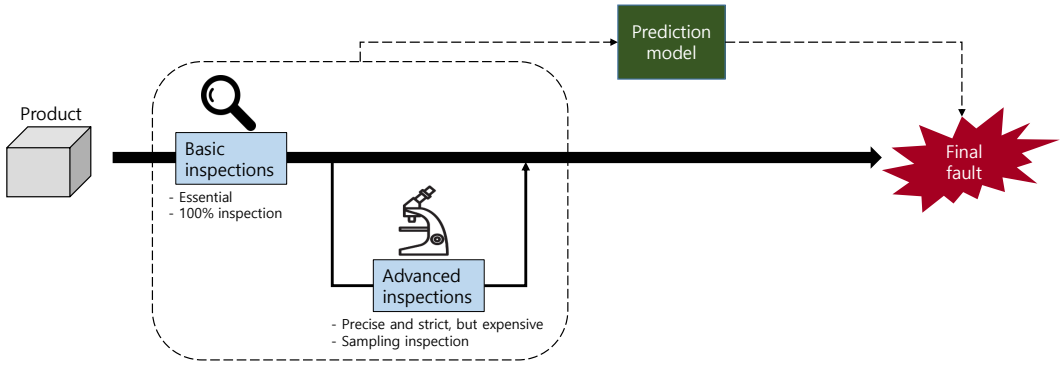


Figure 5.1: Problem situation addressed in Chapter 5.

are conducted only for some sampled products, two different prediction models are constructed: basic and advanced models. The basic model uses only basic inspection results as input variables, while the advanced model uses both basic and advanced inspection results as input variables. Each model is selectively applied depending on whether the product has undergone advanced inspections. The prediction performance of advanced model is relatively high because it uses more information to predict the fault.

In this situation, an increase in the proportion of products that are subject to advanced inspections would lead to an improvement in the fault prediction performance. However, this also increases inspection cost. In other words, a trade-off between inspection cost and prediction performance exists when predicting faults using inspection results. Random sampling has been conventionally adopted to select products that are subject to advanced inspections. However, the necessity of advanced inspections is different for each product. For some products, basic inspections are sufficient to predict their faults. For some other products, advanced inspections are required to make more accurate predictions. The aim of the study is

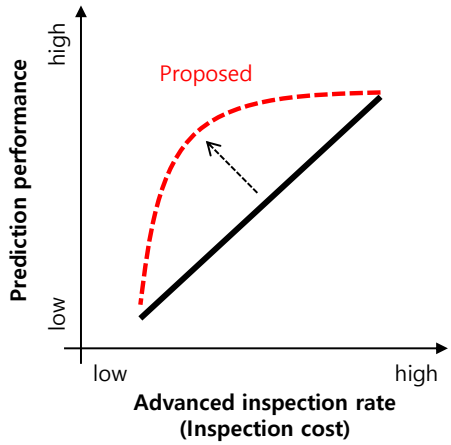


Figure 5.2: Graphical explanation of the purpose of Chapter 5.

to devise an intelligent way of product sampling to obtain a better trade-off between the cost for advanced inspections and the prediction performance, as illustrated in Figure 5.2.

In this chapter, we propose an active inspection framework to predict faults in a cost-effective manner. Assuming that all the inspection results can be secured in the training phase, this framework is about how to operate the prediction models for new products in the prediction phase. In the first step, basic inspections are conducted for all products and the basic model outputs a fault score and its prediction uncertainty score. If the uncertainty score is low, the product's fault score is finalized. If the uncertainty score is high, the product is sent for advanced inspections. The fault score is updated using the advanced model with more inspection results. Through this framework, we can obtain improved fault prediction performance with lower inspection costs. Hence, a better trade-off between inspection cost and fault prediction performance can be realized. Furthermore, we also propose an imputation-based active feature-value acquisition method, termed as expected pre-

diction change (EPC), to obtain a better uncertainty score that can improve the performance of the framework. The effectiveness of the proposed method is demonstrated through a case study using real-world data collected from a semiconductor manufacturer.

In the problem setting of this study, there are two categories of inspections, namely basic inspections and advanced inspections. Basic inspections correspond to necessary inspections that every product should undergo. After basic inspections are completed, advanced inspections are performed. In contrast to the basic inspections, advanced inspections correspond to additional inspections that only selected products undergo. We consider constructing a prediction model to predict whether a product is finally determined as faulty.

We assume that there are n basic inspection items corresponding to input variables $X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$, and m advanced inspection items corresponding to input variables $X_1^{adv}, X_2^{adv}, \ldots, X_m^{adv}$. The output variable Y in training data is 1 for faulty products and 0 for normal products. After the prediction models are constructed, in the prediction phase to make prediction on a new product, the fault score $\hat{Y}$ is yielded with the inspection results. The value of $\hat{Y}$ ranges from 0 to 1.

We focus on the prediction phase, and thus it is assumed that the prediction models can be constructed with the complete training set filled with all inspection results. When operating the models in the prediction phase, inspection results are selectively acquired. All products undergo basic inspections, and thus values for $X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$ can be secured without missing values. Conversely, advanced inspections are only conducted for some sampled products, and thus the values of $X_1^{adv}, X_2^{adv}, \ldots, X_m^{adv}$ can only be obtained for a few selected products.

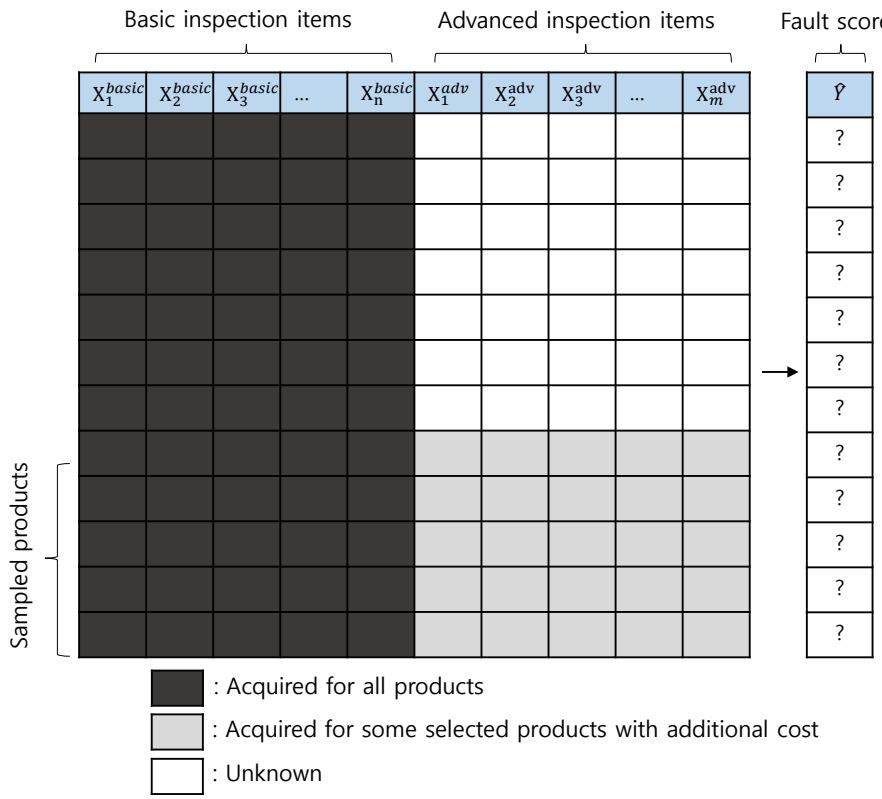


Figure 5.3: Problem situation in terms of data.

The situation is summarized in Figure 5.3. Each instance denotes each product, and each has inspection results as features. Black and gray boxes represent values obtained for all products and values obtained for some selected products, respectively. White boxes represent unknown values.

The goal of this chapter is to maximize the fault prediction performance by effectively selecting a set of products to obtain additional m features of advanced inspection results among the total set of products to be predicted. Thus, we apply active feature-value acquisition to fault prediction.

5.2 Proposed Method

5.2.1 Active inspection framework

We propose an active inspection framework to improve the fault prediction performance in a cost-effective manner. For some products, basic inspections are sufficient to predict their faults and thus do not necessitate advanced inspections. However, for some products, their faults can be predicted more accurately with advanced inspections. Considering the different characteristics of individual products, the products to acquire advanced inspection results are selected by the framework. The procedure of the framework is shown in Figure 5.4.

In the training phase, two fault prediction models are trained, namely basic and advanced models. The basic model F^{basic} predicts the fault with only basic inspection items. The fault score $\hat{Y}^{basic}$ is calculated as a function of $X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$. The advanced model F^{adv} predicts the fault with both basic and advanced inspection items. The $\hat{Y}^{adv}$ is calculated as a function of $X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$ and $X_1^{adv}, X_2^{adv}, \ldots, X_m^{adv}$. The advanced model may perform better than the basic model because the advanced model utilizes more information to make predictions.

In the prediction phase, all products undergo basic inspections to obtain values for the variables $X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$. Then, the fault prediction is conducted by using the basic model F^{basic}. The uncertainty score S representing the prediction uncertainty is also calculated from F^{basic}. Only for products with the score S above a certain threshold τ, advanced inspections are conducted to obtain values of additional variables $X_1^{adv}, X_2^{adv}, \ldots, X_m^{adv}$. Then, fault prediction is conducted using the advanced model F^{adv} for the selected products. Consequently, the fault score $\hat{Y}$ is expressed as follows:

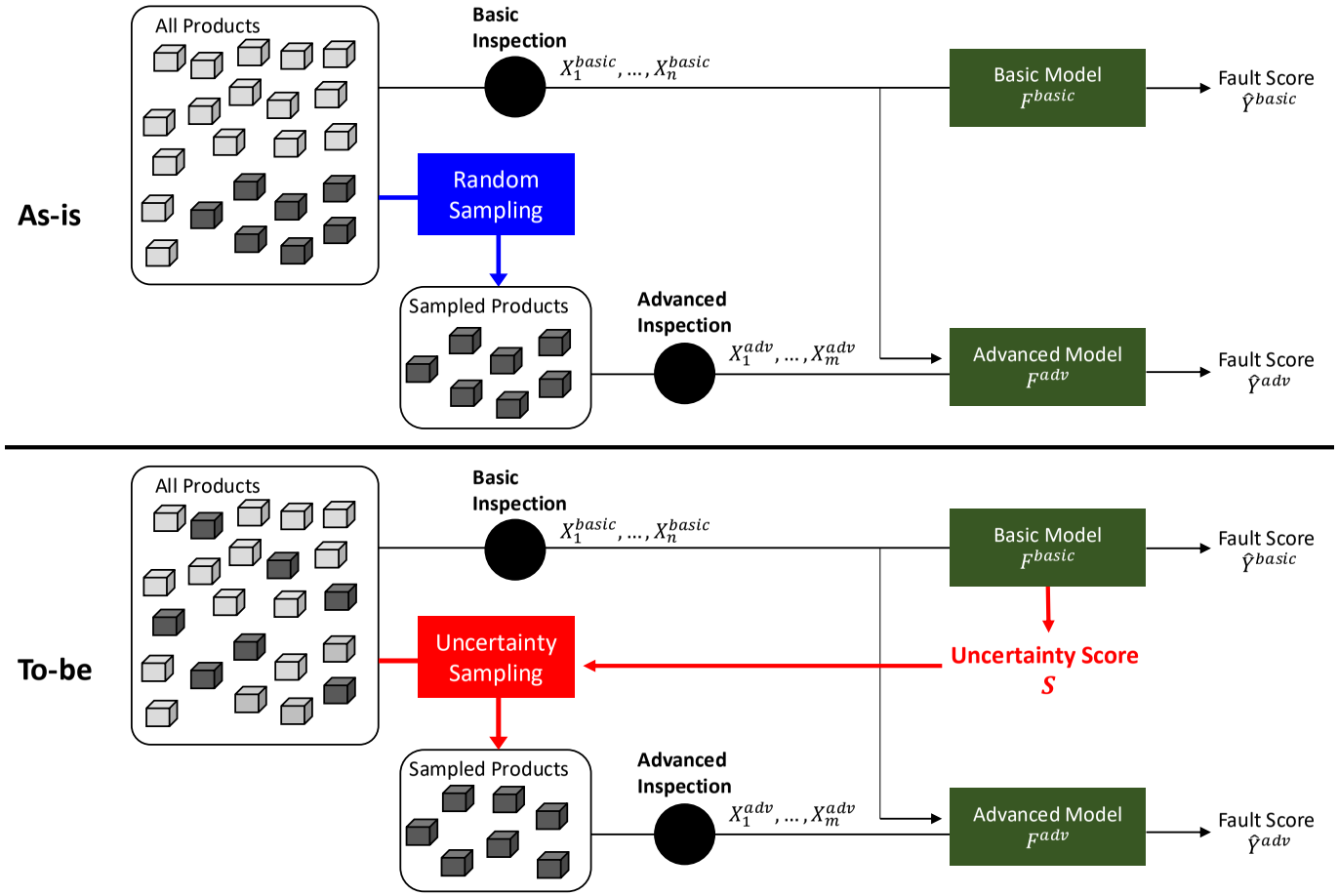


Figure 5.4: Active inspection framework.

$$\hat{Y} = \begin{cases} F^{basic}(X_1^{basic}, \ldots, X_n^{basic}), \text{ if } S \leq \tau; \\ F^{adv}(X_1^{basic}, \ldots, X_n^{basic}, X_1^{adv}, \ldots, X_m^{adv}), \text{otherwise.} \end{cases} \tag{5.1}$$

Choosing an appropriate uncertainty scoring method is critical to the performance of the active inspection framework. To calculate the uncertainty score S, several conventional methods can be used, including 'variance', 'margin', and 'bias'.

- 'variance': The variance of the fault scores $\hat{Y}^{basic}$ calculated by multiple basic models is used [55]. Given k different basic models, k fault scores $\{\hat{Y}^{basic}(i)\}_{i=1}^{k}$ are calculated, and then the uncertainty score S is calculated as:

$$S = \frac{\sum_{i=1}^{k} (\hat{Y}^{basic}(i) - \overline{\hat{Y}^{basic}})^2}{k-1} \tag{5.2}$$

- 'margin': The inverse of the absolute difference between the fault score $\hat{Y}^{basic}$ and normal score $1 - \hat{Y}^{basic}$ is used [70]. This method is similar to selecting instances whose fault scores are close to 0.5. The uncertainty score S is:

$$S = \frac{1}{|\hat{Y}^{basic} - 0.5|} \tag{5.3}$$

- 'bias': Considering class imbalance of the training dataset, it calculates the proximity between the fault score $\hat{Y}^{basic}$ and fault rate FR in the training dataset. Specifically, an uncertainty score S is calculated as follows:

$$S = \begin{cases} \dfrac{\hat{Y}^{basic}}{FR}, & \text{if } \hat{Y}^{basic} \leq FR; \\ \dfrac{1 - \hat{Y}^{adv}}{1 - FR}, & \text{otherwise,} \end{cases} \tag{5.4}$$

Table 5.1: Description of models used in EPC.

	model	input	output
imputation models	F_1^{imp}	$X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$	$\hat{X}_1^{adv}$
	F_2^{imp}	$X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$	$\hat{X}_2^{adv}$
	$\vdots$	$\vdots$	$\vdots$
	F_m^{imp}	$X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$	$\hat{X}_m^{adv}$
fault prediction models	F^{basic}	$X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$	$\hat{Y}^{basic}$
	F^{adv}	$X_1^{basic}, \ldots, X_n^{basic}, \hat{X}_1^{adv}, \ldots, \hat{X}_m^{adv}$	$\tilde{Y}^{adv}$

The value of S is maximum when $\hat{Y}^{basic} = FR$ and is minimum when $\hat{Y}^{basic} =$ 0 or 1.

The above-mentioned methods only focuses on the uncertainty of the basic model. However, high uncertainty of the basic model does not necessarily indicate that the advanced model can complement the basic model. To overcome the limitation, we present a novel uncertainty scoring method that is appropriate for the framework in the next subsection.

5.2.2 Acquisition based on Expected Prediction Change

To obtain the uncertainty score S in the active inspection framework, we propose an imputation-based feature-value acquisition method, named EPC. EPC directly uses the advanced model to approximate the change in prediction. The scheme of the method is shown in Figure 5.5. The models used in the method are summarized in Table 5.1.

In the training phase, in addition to the basic model and advanced model, we also train imputation models. These models are used to estimate the advanced inspection results, $X_1^{adv}, X_2^{adv}, \ldots, X_m^{adv}$, through basic inspection results, $X_1^{basic}, X_2^{basic}, \ldots, X_n^{basic}$. That is, they consist of m regression models, $F_1^{imp}, F_2^{imp}, \ldots, F_m^{imp}$, and each model

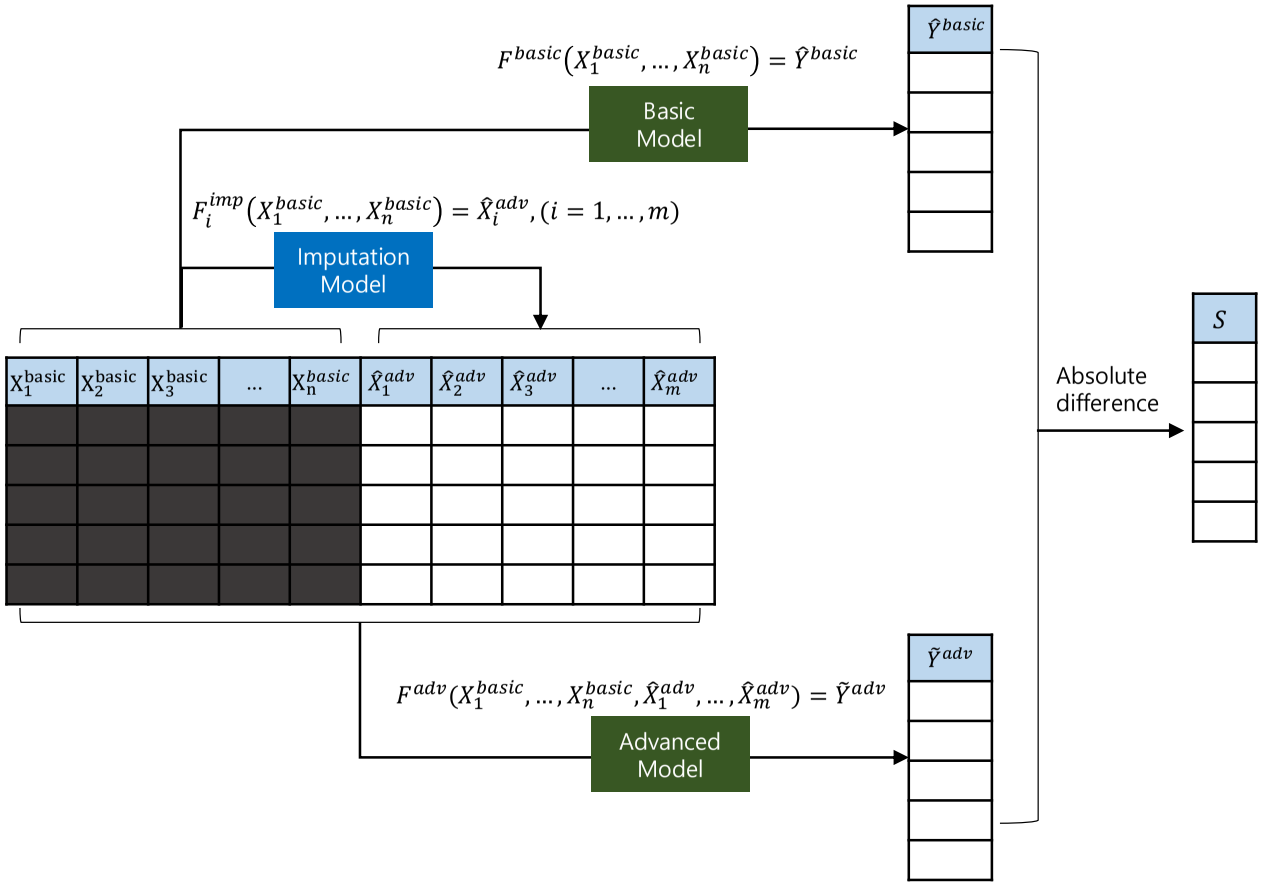


Figure 5.5: Illustrative description of EPC.

estimates m advanced inspection results. Imputation models are expressed as follows:

$$\hat{X}_i^{adv} = F_i^{imp}(X_1^{basic}, \ldots, X_n^{basic}), \quad i = 1, \ldots, m \tag{5.5}$$

In the prediction phase, after basic inspections are conducted, two different fault scores are calculated as follows. For the first fault score, $\hat{Y}^{basic}$ is calculated by simply applying the basic model on the basic inspection results. For the other fault score, $\tilde{Y}^{adv}$ is calculated, which is the expected fault score when advanced inspection results are obtained. Estimated advanced inspection results, $\hat{X}_1^{adv}, \hat{X}_2^{adv}, \ldots, \hat{X}_m^{adv}$, are calculated using imputation models. Subsequently, the advanced model yields the fault score, $\tilde{Y}^{adv}$, with the basic inspection results and estimated advanced inspection results. The two fault scores are expressed as follows:

$$\begin{aligned} \hat{Y}^{basic} &= F^{basic}(X_1^{basic}, \ldots, X_n^{basic}) \\ \tilde{Y}^{adv} &= F^{adv}(X_1^{basic}, \ldots, X_n^{basic}, \hat{X}_1^{adv}, \ldots, \hat{X}_m^{adv}) \end{aligned} \tag{5.6}$$

To select products to undergo advanced inspections, the uncertainty score S is defined as the absolute difference between $\hat{Y}^{basic}$ and $\tilde{Y}^{adv}$ as below:

$$S = |\hat{Y}^{basic} - \tilde{Y}^{adv}|. \tag{5.7}$$

Products with the high uncertainty score S are selected for advanced inspections. A higher value of S implies that outputs between basic and advanced models are expected to be more different. Hence, the advanced model is expected to complement

the basic model.

5.3 Experiments

5.3.1 Data description

We conducted a case study on real-world data obtained from a semiconductor manufacturer based in the Republic of Korea to investigate the effectiveness of the proposed methods. In semiconductor manufacturing, a wafer test is conducted after wafer fabrication to test electrical properties for each die in a wafer. The wafer test consists of a number of inspections, which can be divided into basic and advanced inspections. Advanced inspections are time-consuming and costly because they consist of rigorous examination under severe conditions. Hence, they are performed for a few sampled dies in a wafer. The wafer test is followed by the assembly step to fabricate final products. Finally, the final test is conducted to determine whether each product passes or fails through the functionality evaluation. The purpose of the case study was to predict failures in the final test using wafer test results.

We used two datasets for which the descriptions are listed in Table 5.2. The two datasets correspond to different inspection recipes (`recipe1`, `recipe2`). The basic inspection items are identical for both recipes. `Recipe1` includes more advanced inspection items albeit fewer instances, while `recipe2` includes fewer advanced inspection items albeit more instances. The inspection items correspond to input variables of the dataset. We also obtain output variable indicating as to whether each product is faulty in the final test. The datasets are highly imbalanced with a fault rate of less than 1%.

Table 5.2: Description for dataset used in Chapter 5.

inspection recipe	n. instances	n. input variables	
		basic inspection	advanced inspection
`recipe1`	18,471	8	91
`recipe2`	245,006	8	80

5.3.2 Fault prediction models

We conducted preliminary experiments to select the learning algorithm for constructing fault prediction models. The effectiveness of the three learning algorithms was evaluated : random forest (RF), artificial neural network (ANN), and k-nearest neighbor (kNN). For RF, the number of trees was set to 500, and the minimum number of instances to split for each tree was set to 10. With respect to ANN, two hidden layers with 50 tanh units in each were used. With respect to kNN, the number of neighbors k was set to 50. All classifiers were implemented using scikit-learn package and all other settings were set as the defaults for the package.

The performance of each classifier was evaluated in terms of the area under the receiver operating characteristic curve (AUROC) [6] via the two-fold cross validation procedure. In the procedure, we randomly split the dataset into two-fold, namely one for training and the other for test, and vice versa. The experiments were repeated 30 times independently.

Table 5.3 lists the average and standard deviation of the performance of the compared classifiers. As shown in Table 5.3, RF exhibited high and stable performance on both datasets. This is in line with the original paper [7] that RF is robust to noise and outliers of input variables. Based on the results of the preliminary experiments, we selected RF to construct fault prediction models for the proposed framework.

Table 5.3: Preliminary experiment results in AUROC. (mean ± standard deviation)

		RF	ANN	kNN
`recipe1`	basic model	**0.7132 ± 0.0278**	0.6970 ± 0.0790	0.6510 ± 0.0323
	advanced model	**0.7477 ± 0.0308**	0.7180 ± 0.0375	0.6139 ± 0.0302
`recipe2`	basic model	**0.6957 ± 0.0096**	0.6890 ± 0.0244	0.6226 ± 0.0080
	advanced model	**0.7347 ± 0.0093**	0.7137 ± 0.0193	0.6274 ± 0.0081

5.3.3 Experimental design

In experiments, we simulated the proposed active inspection framework. As imputation models which estimate advanced inspection results with basic inspection results, three algorithms were investigated: DTs, RFs, and extremely randomized trees (ETs). That is, three versions of EPC are implemented: EPC_DT, EPC_RF, and EPC_ET, respectively.

Three conventional uncertainty scoring methods were used as the baseline to obtain the uncertainty score: 'variance', 'margin', 'bias'. In case of 'variance', since RF itself is an ensemble based method, the variance of fault scores calculated by each tree in an RF was utilized for the uncertainty score. Additionally, random sampling was also compared as a baseline.

For all the tree-based models, the minimum number of instances to split was set to 10. The RFs used as the basic and advanced models consisted of 500 trees. The RFs and ETs used as the imputation models consisted of 50 trees. All models were implemented using scikit-learn package and all other hyperparameters were set as the defaults of the package.

After all the models were trained using the training set, the proposed active inspection framework was simulated. We examined the performance with varying inspection costs. The proportion of products undergoing advanced inspections among

all products in the test set, advanced inspection rate, was varied from zero to one at 0.1 intervals, which was controlled through the threshold τ for uncertainty score. The advanced inspection rate of zero indicates that all products undergo only the basic inspections whereas the advanced inspection rate of one implies that all products are subject to advanced inspections.

The performance of each method was evaluated in terms of AUROC via two-fold cross validation. All experiments were repeated 50 times independently. We report their average in the next section.

5.3.4 Results and discussion

Figure 5.6 shows the comparison results. The x-axis denotes the advanced inspection rate to sample the products that are examined with advanced inspections. The inspection cost is proportional to the advanced inspection rate. The y-axis denotes the AUROC for fault prediction performance through the framework. The upward trend indicates the trade-off in which the fault prediction performance tends to improve when total inspection costs increase. The graph that are closer to the top left indicates a better trade-off, *i.e.*, higher performance with lower inspection cost.

The random sampling method increased the prediction performance in an almost linear manner with inspection costs. In comparison, 'variance', 'margin', and 'bias' method showed a better trade-off than the random sampling. In both datasets, 'margin' exhibited almost the same performance as 'variance', and 'bias' method revealed some additional improvement by considering the class imbalance problem. The results indicated that the active inspection framework performs well.

In both datasets, the proposed EPC method outperformed baseline methods. Specifically, a large performance difference was observed between data acquisition

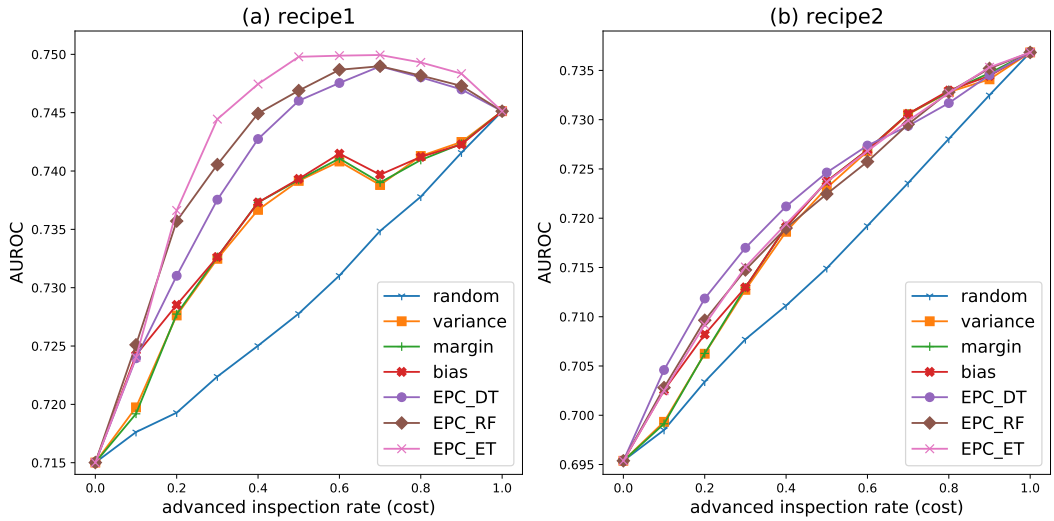


Figure 5.6: Experimental results for each dataset.

methods in `recipe1` because it includes more advanced inspection items and relatively fewer instances. Even when the advanced inspection rate exceeded 0.5, EPC exhibited higher AUROC than that of 100% advanced inspection irrespective of the types of imputation models. This means that we can achieve a prediction performance comparable to that of 100% advanced inspection even if only 50% of all products undergo advanced inspections. In the case of `recipe2`, EPC_DT exhibited the best performance, especially when the advanced inspection rate was low. The result is considerably meaningful given that the advanced inspection rate in reality is not high.

The experimental results demonstrated that the proposed method obtains higher fault prediction performance at lower cost. It is conjectured that instances for advanced inspections are selected in an optimal manner by the proposed method.

Chapter 6

Adaptive Fault Detection for Recipe Transition

6.1 Problem Description

In semiconductor manufacturing, yield is one of the most important index which affects the manufacturer's profitability. To achieve high yield, manufacturers make efforts to detect wafer faults at an early manufacturing step and prevent the faulty wafers from proceeding to the downstream steps. With the growth of Industry 4.0, a lot of sensors on equipment collect tremendous data to describe the status including temperature, pressure, and gas flow. Recently, this multi-dimensional time series sensor data has been utilized as input of a machine learning model for fault detection and classification (FDC) to predict whether each wafer is faulty or not in the future.

As shown in Figure 6.1, an FDC model is constructed by training a prediction model using sensor data at a specific process step as input and inspection results (faulty or normal) for the wafers that have completed the entire fabrication as output. This kind of data-driven modeling assumes that training data and test data are drawn from the same distribution. Therefore, when training the model, a lot of training data consistent with test data is required. Moreover, especially when training a deep learning models, the amount of data required becomes larger. However, acquiring consistent data for training FDC model is almost impossible in real-world

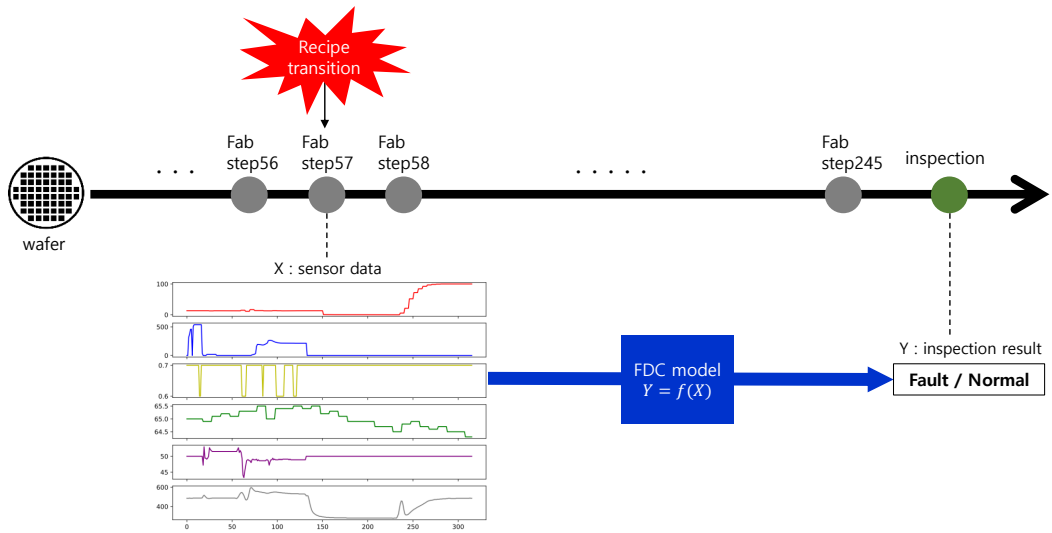


Figure 6.1: Illustrative description of the problem situation to be addressed in the Chapter 6.

semiconductor manufacturing process because the transition of recipe occurs frequently. Recipe in manufacturing refers to a collection of setting values for a fabrication process such as the amount of each ingredient, duration of the process, and the temperature setting. To improve the manufacturing quality, the recipe changes frequently, especially in the production of newly developed products [4]. The transition of recipe causes a change in characteristics of the input sensor data. Therefore, the FDC model trained with data from old recipe may have bad generalization performance for the data from new recipe. Whenever a recipe transition occurs, it is necessary to retrain the FDC model with the new recipe data, but it takes time to secure a lot of labeled training data for the new recipe.

In this chapter, we propose adaptive fault detection framework for transition of recipe. The objective of this framework is to minimize the performance degradation caused by the recipe transition. Initially, the sensor data of the wafers made with

old recipe and the final inspection data of these wafers are given, and the FDC model is constructed with these data. Immediately after the recipe transition occurs, input sensor data from the new recipe can be obtained, while wafer inspection data cannot be obtained until the whole fabrication of the wafer is completed. For this situation, unsupervised domain adaptation method is employed to reduce the performance degradation for new recipe. After the fabrication process is completed and output data is acquired for some wafers, semi-supervised domain adaptation is employed to compensate for still small amounts of labeled data from new recipe. Through experiments using real-world data from a semiconductor manufacturer, we demonstrated that the proposed framework can reduce the performance degradation caused by recipe transition.

6.2 Proposed Method

6.2.1 Overview

Wafer fabrication process consists of hundreds of process steps. Wafers completed through the entire process steps are inspected to determine whether they are faulty. The FDC model is a prediction model that predicts whether a wafer will become faulty after it is completed, by using sensor data at a specific process step as input. Therefore, model training requires two kinds of dataset corresponding to input and output respectively: sensor data at a specific process step for each wafer and final inspection result data for the wafers. In this chapter, FDC models are constructed using the self-attentive CNN structure of Kim *et al.* [40].

An overview of the proposed framework is shown in Figure 6.2. It is assumed that the sensor data from an input process step with a specific recipe and the cor-

responding inspection result data are given, and an FDC model trained with them is also given. We will refer to this recipe and the model as the old recipe and the old model. When a recipe transition occurs, wafers begin to go through the input process step with a new recipe, and the distribution of the sensor data become different from before. Therefore, using the old model as it is leads to significant performance degradation. At this point, the proposed framework can be used. The wafer fabrication process consists of a huge number of process steps, and a turnaround time (TAT) for it is quite long, so there is a time interval between acquiring input data and output data. Therefore, the proposed framework can be divided into two phases, unsupervised adaptation phase and semi-supervised adaptation phase. The unsupervised adaptation phase corresponds to the period from the time when the recipe transition occurs to the time before the inspection results of the new recipe wafers appear. In this phase, the sensor data for the new recipe exists, but the corresponding inspection result data does not exist. The semi-supervised adaptation phase corresponds to the period after a new recipe wafer is completed and the inspection results begin to appear. In this phase, the sensor data for the new recipe exists, and the corresponding inspection result data partially exist. The notations used to describe each phase are summarized in Table 6.1.

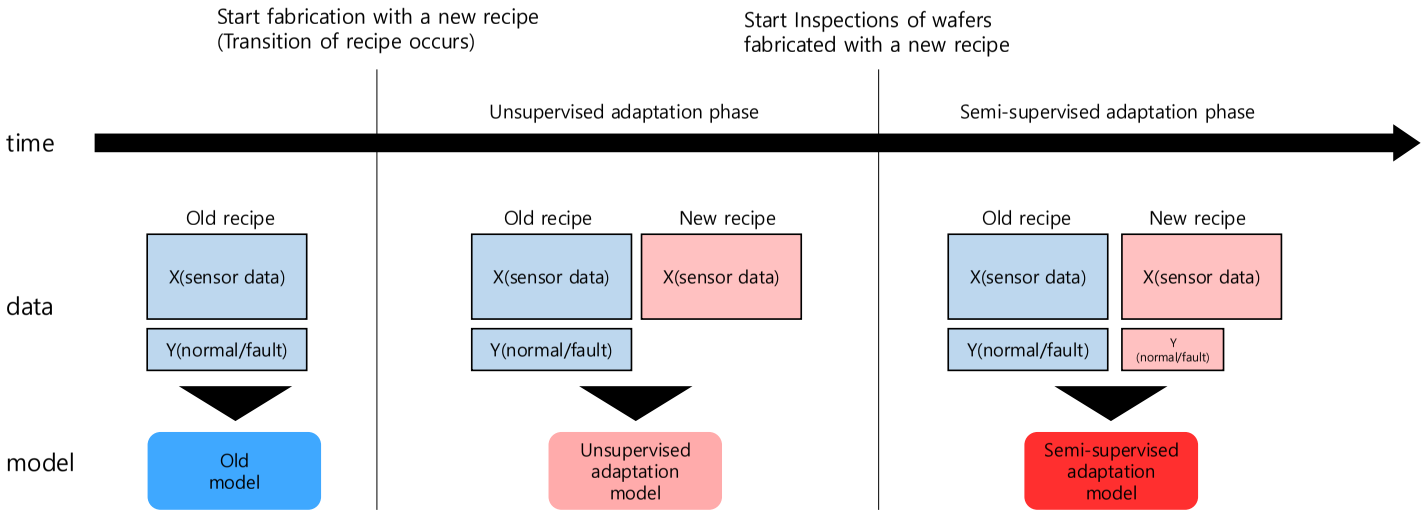


Figure 6.2: Overview of the adaptive fault detection framework for recipe transition.

Table 6.1: Notations used in Chapter 6.

Notation	Type	Description
X^S	set	sensor dataset from old recipe.
Y^S	set	inspection results corresponding to wafers of X^S.
X^T	set	sensor dataset from new recipe.
X_L^T	set	sensor dataset from new recipe with inspection results (labeled).
Y_L^T	set	inspection results corresponding to wafers of X_L^T.
X_U^T	set	sensor dataset from new recipe without inspection results (unlabeled).
g	function	feature extraction network. $g : \mathcal{X} \to \mathcal{Z}$
h	function	classification network. $h : \mathcal{Z} \to \mathcal{Y}$
f	function	FDC model. $f = h \circ g$

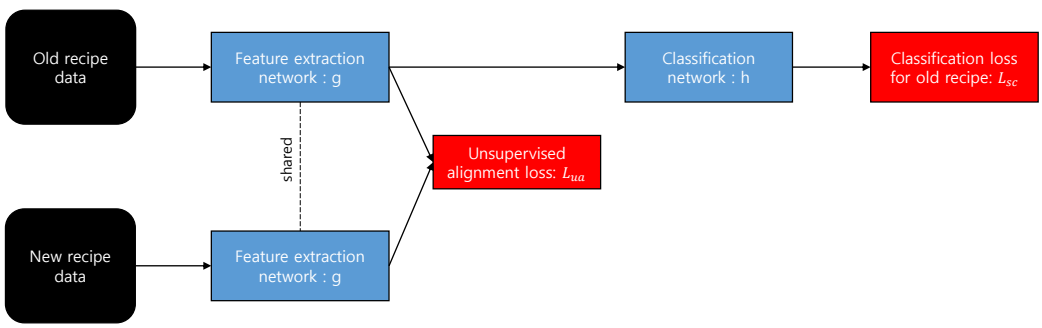


Figure 6.3: Overall neural network architecture used to train the FDC model in unsupervised adaptation phase.

6.2.2 Unsupervised adaptation phase

Unless the input process step in which the input sensor data is generated is the last step of the entire fabrication process, there is a time interval between the process step and the final inspection of a wafer. In other words, it takes a long time to acquire output data after input data is acquired. Before output data is acquired, an FDC model for the new recipe must be built with only input data. To this end, we apply unsupervised domain adaptation method using the old recipe as a source and the new recipe as a target.

In general, deep learning-based FDC model f could be modeled by the composi-

tion of feature extraction function g and classification function h, *i.e.*, $f = h \circ g$. If the feature representations extracted by g for each recipe have a similar distributions, classification function h trained for the old recipe can perform similarly for the new recipe as well. Then, the FDC model f can be used for the new recipe. To align the old recipe data and the new recipe data in the feature space constructed with the g, we adopt a siamese network for training the FDC model. The siamese network consists of two identical networks for each source and target with shared weights. In the unsupervised adaptation setting, classification function h for target data is not required. Overall neural network architecture used to train the FDC model in unsupervised adaptation phase is depicted in Figure 6.3. In the proposed framework, MMD loss is adopted because it can be extended to semi-supervised settings [85]. With the source and target representation calculated by a feature extraction network$g$, MMD is computed as followed:

$$\mathrm{MMD}(X^S, X^T) = \left\| \frac{1}{X^S} \sum_{x^s \in X^S} g(x^s) - \frac{1}{X^T} \sum_{x^t \in X^T} g(x^t) \right\| \tag{6.1}$$

Then, unsupervised alignment loss $\mathcal{L}_{ua}(X^S, X^T)$ can be expressed as

$$\mathcal{L}_{ua}(X^S, X^T) = \mathrm{MMD}^2(X^S, X^T). \tag{6.2}$$

By minimizing this loss, the representations of the source and the target are aligned. With a regular classification loss for source data $\mathcal{L}_{sc}(X^S, Y^S)$, total loss $\mathcal{L}_{total}$ can be expressed as

$$\mathcal{L}_{total} = \mathcal{L}_{sc}(X^S, Y^S) + \lambda \mathcal{L}_{ua}(X^S, X^T). \tag{6.3}$$

The hyperparameter λ determines how strongly align between the recipes. Since $\mathcal{L}_{ua}$

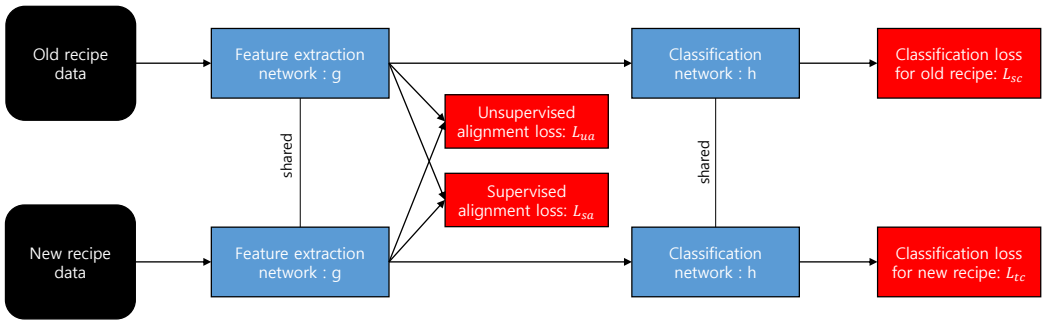


Figure 6.4: Overall neural network architecture used to train the FDC model in semi-supervised adaptation phase.

makes the recipes indistinguishable, after the training ends, we directly apply the FDC model f to the target data, the new recipe data.

6.2.3 Semi-supervised adaptation phase

After a wafer goes through the input process step and enough time passes, the entire fabrication process for the wafer is finished, and then it is subjected to the finial inspection. When the inspection results indicating whether each wafer is normal or faulty begin to appear, they can be used as labels. Because the whole fabrication process is large and complex, even wafers that have gone through the input process step at the similar time have different timings to complete the whole fabrication process. So, the inspection results to be used as labels cannot be obtained at once, but are obtained little by little at time intervals. For this situation, we apply semi-supervised domain adaptation method.

Overall neural network architecture used to train the FDC model in semi-supervised adaptation phase is depicted in Figure 6.4. As in the unsupervised adaptation phase, a siamese network is used to train a FDC model f. Unlike unsupervised adaptation phase, classification loss can be calculated for some labeled target data and super-

vised alignment can also be used in addition to unsupervised alignment.

For still unlabeled target data, unsupervised alignment loss is used in the same way as in the unsupervised adaptation phase.

$$\mathcal{L}_{ua}(X^S, X_U^T) = \mathrm{MMD}^2(X^S, X_U^T) \tag{6.4}$$

Only with the unsupervised alignment loss, there is no guarantee that instances from different recipes but the same class would map nearby in the feature space constructed with g. This can be addressed with supervised alignment loss that aligns separately for each class as followed:

$$\mathcal{L}_{sa}(X^S, X_L^T) = \sum_{c \in \{0,1\}} \mathrm{MMD}^2(X^S(c), X_L^T(c)) \tag{6.5}$$

where $X^S(c) = X^S|\{Y^S = c\}$ and $X_L^T(c) = X_L^T|\{Y_L^T = c\}$. c indicates class, 0 for normal and 1 for fault. In this phase, in addition to the classification loss for old recipe $\mathcal{L}_{sc}(X^S, Y^S)$, classification loss for new recipe $\mathcal{L}_{tc}(X_L^T, Y_L^T)$ can be calculated by using some labeled target data. Finally, total loss $\mathcal{L}_{total}$ can be presented as follow:

$$\mathcal{L}_{total} = \mathcal{L}_{sc}(X^S, Y^S) + \mathcal{L}_{tc}(X_L^T, Y_L^T) + \lambda(\mathcal{L}_{ua}(X^S, X_U^T) + \mathcal{L}_{sa}(X^S, X_L^T)). \tag{6.6}$$

As in the unsupervised adaptation phase, the hyperparameter λ determines how strongly align between the recipes. After the training ends, the FDC model f can be applied to the target data.

6.3 Experiments

6.3.1 Data description

We conducted experiments on real-world data provided by a semiconductor manufacturing company in South Korea to investigate the effectiveness of the proposed framework. Two sensor datasets with two different recipes, recipe 1 and recipe 2, were collected from a specific etch process step. Each recipe dataset contains 778 and 1,546 wafers, respectively. All sensor values were recorded every 0.1 second. The step in which the sensor data is collected consists of 25 inner steps, among which domain engineers selected a set of consecutive steps for fault detection. Then, the lengths of the selected sequences are 315.6 and 315.3 on average for recipes 1 and 2, respectively. Among hundreds of sensors installed in the equipment, 65 sensors were used for analysis after excluding the sensors that have the same values for the selected time length. In addition, inspection results for the wafers were collected after the whole fabrication process is completed. They indicate whether each wafer is normal or faulty. There exist 22 and 31 faulty wafers in recipes 1 and 2, respectively, showing severe class imbalance of 2.8% and 2.0% fault rate.

6.3.2 Experimental setting

In the experiments, self-attentive CNN from Kim *et al.* [40] was adopted as an FDC model structure in the proposed framework. In this model, to directly deal with sensor data with different time lengths for each wafer, the self-attention mechanism [50] is applied after the CNN structure having convolutional filters that move along the time axis. This self-attention mechanism converts variable-length sensor data into a fixed-size vector. As shown in the original paper, this self-attention-based

model makes it possible to diagnose the cause of the fault through the distribution of attention weights. This self-attentive CNN obtained high fault detection performance even with a short training time. This model consists of one convolutional layer, one self-attention layer, and two fully connected layer. In this experiment, the last one fully connected layer was considered as a classification network h, and all previous layers were considered as a feature extraction network g.

When training a model, we used the Adam optimizer with a learning rate of 0.001 and a batch size of 256. All models were trained for 200 epochs. The hyperparameter λ in Equation (6.3) and Equation (6.6) was set to 0.01. When performing the recipe adaptation, all weights of the network were initialized with that of the old model.

We conducted experiments by simulating a situation where a recipe transition occurred. With the two datasets, experiments were performed for the transitions in two directions: from recipe 1 to recipe 2 and recipe 2 to recipe 1. The performance of the fault detection was evaluated in terms of AUROC via three-fold cross validation. All experiments were repeated 30 times independently and their average is reported.

6.3.3 Performance degradation caused by recipe transition

We demonstrated the performance degradation that occurs when the old model constructed with the old recipe data is applied to the new recipe data as it is. After an FDC model was constructed with the data of one recipe, we applied the model to the test data of the same recipe and the other recipe, respectively. The comparison results are shown in Table 6.2. As shown in the table, fault detection performance decreased significantly when the model was applied to the different recipe from the recipe used for training. This suggests that recipe transition reduces the performance of the existing FDC model, and therefore the model need to be retrained for the

Table 6.2: Performance degradation when recipe transition occurs.

	test on the same recipe	test on the other recipe
model trained with recipe 1	0.7886	0.5305
model trained with recipe 2	0.9401	0.6097

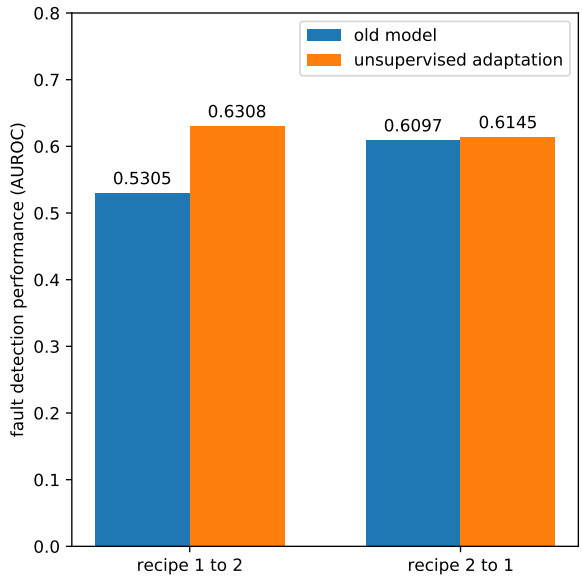


Figure 6.5: Comparison of the FDC models for unsupervised adaptation phase.

new recipe.

6.3.4 Effect of unsupervised adaptation

First, we identified the effect of the unsupervised adaptation phase. The unsupervised adaptation model and the old model were applied to a separate test dataset for the new recipe, respectively, and the comparison results are shown in Figure 6.5. In the both recipe transition situations, the unsupervised adaptation model showed improved performance compared to the old model. This suggests that before the inspection results of the new recipe wafers appear, the alignment of feature rep-

resentations by unlabeled data can reduce the performance degradation caused by recipe transition. The effect of unsupervised adaptation was relatively greater at the transition to recipe 2, where the amount of data is higher than that of recipe 1. It can be conjectured that this is because the amount of unlabeled data in the new recipe is important to the effect of the alignment.

6.3.5 Effect of semi-supervised adaptation

To identify the effect of the semi-supervised adaptation phase, we conducted experiments by gradually increasing the proportion of labeled wafers in the new recipe data to 10%, 20%, 40%, 70%, 100%. In addition to the proposed semi-supervised adaptation model, a joint training model and a target model were also constructed as baseline models. The joint training model was trained only with the classification loss for old recipe and new recipe without any alignment losses, and the target model was trained only with the classification loss for new recipe through an original single structure rather than a siamese structure. These three models constructed were applied to a separate test dataset for the new recipe.

The comparison results are shown in Figure 6.6. As expected, it can be seen that the fault detection performance of all models increases as the proportion of labeled wafers increases, that is, as the inspection for the new recipe wafers progress. The target model showed the lowest performance among the three models, and was particularly poor when the proportion of labeled wafers was low. The joint training model showed a great improvement in performance by using old recipe data as well as new recipe data. The semi-supervised adaptation model achieved additional performance improvement by performing alignment using unlabeled data of new recipe. In particular, the improvement was significant when the proportion of labeled

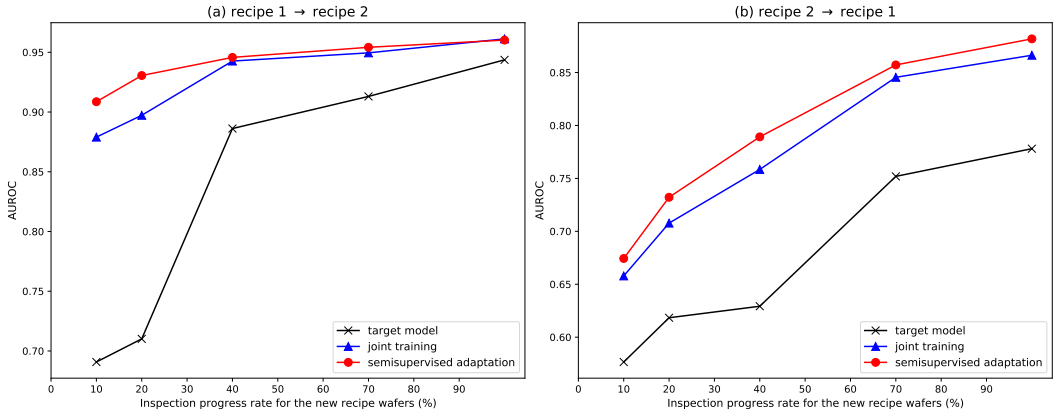


Figure 6.6: Comparison of the FDC models as inspection rate increases.

wafers was low. This suggests that the performance can be recovered more quickly by using semi-supervised adaptation.

As more labeled wafers for new recipe are secured, it become possible to achieve high fault detection performance by using them alone without other source data, reducing the difference in performance between methodologies. This trend was clear in the transition from recipe 1 to 2. Since the total amount of recipe 1 data is relatively small, this trend was not clearly seen in the transition from recipe 2 to 1. If the amount of recipe 1 data is increased, it can be inferred that the same trend will be observed.

In the transition from recipe 1 to 2, it is particularly noteworthy that the performance of the semi-supervised adaptation model when only 10% of the new recipe wafers were labeled was comparable to that when 100% was labeled. This demonstrates that the semi-supervised adaptation can improve the performance of the FDC model when only a small amount of inspection results for new recipe wafers are obtained after the recipe transition occurs.

As for the comprehensive explanation based on all the previous experimental results, the performance degradation of the FDC model caused by recipe transition can be reduced by unsupervised adaptation, and the resilience to recipe transition can be improved by semi-supervised adaptation. That is, through the proposed framework, it is possible to adapt quickly to changes in distribution caused by recipe transition.

Chapter 7

Conclusion

7.1 Contributions

Predictive modeling is used in various aspects in manufacturing systems, such as automation of visual inspection, prediction of faulty products, and result estimation of expensive inspection. When constructing such a predictive model, sufficient data is essential to achieve high prediction performance. However, in manufacturing systems, it is practically impossible to acquire enough data of all kinds that are needed for the predictive modeling. The difficulty in data acquisition is largely due to three reasons: labeling cost, inspection cost, and environmental change. Labeling task that must be done by experienced engineers is costly, and all inspection processes incur cost. Acquiring the values of both the input variable and the output variable incurs time and monetary costs. Because there are time and monetary constraints in the manufacturing system, it is impossible to obtain all the desired data. In addition, a change in the manufacturing environment make the data acquisition difficult. The manufacturing environment constantly changes. This change transforms the distribution of generated data, making it impossible to obtain enough consistent data. Then, the performance of an existing predictive model is degraded, and the model needs to be retrained with new data. Therefore, a new model has to be created with

a small amount of data.

We solved the labeling cost in wafer map pattern classification, inspection cost in fault detection, and recipe transition in FDC model, respectively. For wafer map pattern classification, we proposed a cost-effective wafer map pattern classification system through the active learning of a CNN. In the system, a CNN model is constructed based on four main steps: uncertainty estimation, query wafer selection, query wafer labeling, and model update. By repetitively performing these steps, a CNN model can be constructed with a higher classification performance and a lower labeling cost. We carried out experimental comparisons between various uncertainty estimation methods for the CNN. The proposed method improved the classification performance compared to the baselines. We found that BALD and mean-STD with diversified top-K selection perform better for the proposed system.

In addition, an active cluster annotation was also proposed to improve the above active learning framework. After clustering was conducted on the unlabeled wafer maps, all the steps, such as uncertainty estimation, query selection, and annotation, were conducted at the cluster-level. Using this method, a better CNN can be obtained with reduced annotation cost. Additionally, we discovered that cluster-level annotation is a robust annotation method that can yield consistent labels.

For the inspection cost for fault prediction problem, we proposed a active inspection framework. Due to the high cost of inspection, advanced inspections are conducted as a sampling inspection. In the situation, a trade-off between inspection cost and prediction performance exists. with a cost-effective active inspection framework, we can obtain a better trade-off. In the framework, the basic model and advanced model are used. Products selected by uncertainty score are subject

to advanced inspections to maximize prediction performance. We also proposed an imputation-based active feature-value acquisition method termed as EPC to improve the framework.

For FDC model with recipe transition, adaptive fault detection framework was proposed. After a transition of recipe occurs, it takes enough time to secure a lot of labeled training data for the new recipe. Adaptive fault detection framework can minimized performance degradation due to transition of recipe. In this framework, immediately after the recipe transition occurs, unsupervised adaptation is employed to reduce the performance degradation, and after inspection results for some new recipe wafers are acquired, semi-supervised adaptation is employed to quickly recover from the performance degradation. Through experiments using real-world data, we demonstrated that the proposed framework can minimize the performance degradation of the FDC model caused by recipe transition.

The contribution of this dissertation can be summarized as follows. First, we specified the data acquisition problem that occurs when conducting predictive modelings in the manufacturing system. Second, we proposed methodologies to solve each problem using active learning, active feature-value acquisition, and domain adaptation. Third, we conducted experiments on real-world data and demonstrated the effectiveness of each methodology.

In this dissertation, we applied and improved active learning, active feature-value acquisition, and domain adaptation to overcome data acquisition problems in wafer map pattern classification, fault prediction, and FDC model, respectively. Although some specific predictive modeling problems have been addressed and improved here, these methodologies can be similarly applied throughout predictive modelings in

manufacturing systems. The proposed active learning framework could certainly be used for other visual inspection automation, and active inspection could be applied to almost any manufacturing inspection process. In addition, the adaptive fault detection methodology can be applied to environmental changes other than recipe transitions, and of course, it can be used in all manufacturing fields other than semiconductor manufacturing.

7.2 Future work

As future work, we will investigate and improve other methodologies to solve the data acquisition problems. More specifically, combining the active learning with a semi-supervised learning scheme would improve the effectiveness through exploiting abundant unlabeled data. Combination of active learning and active feature-value acquisition can also be considered in the situation where collecting values of both input and output variables is costly. In addition, self-correction methods will be investigated to reduce noisy labels, through which training data can be annotated more accurately. In the case of the adaptive fault detection model, we plan to further improve it by applying other state-of-the-art domain adaptation algorithms [67, 87, 60], and study predictive models that can adapt to other factors that cause changes in input data distribution, such as a preventive maintenance of equipment. An adaptive model that can utilize multiple source domains will also be developed for a situation where the period of change in data distribution is much shorter. Finally, research on directly injecting domain knowledge into data-driven models will have great significance in compensating for data insufficiency.

Bibliography

[1] F. Adly, O. Alhussein, P. D. Yoo, Y. Al-Hammadi, K. Taha, S. Muhaidat, Y.-S. Jeong, U. Lee, and M. Ismail, *Simplified subspaced regression network for identification of defect patterns in semiconductor wafer maps*, IEEE Transactions on Industrial Informatics, 11 (2015), pp. 1267–1276.

[2] J. Azimi, A. Fern, X. Z. Fern, G. Borradaile, and B. Heeringa, *Batch active learning via coordinated matching*, in Proceedings of International Conference on Machine Learning, 2012, pp. 307–314.

[3] S. H. Bang, R. Ak, A. Narayanan, Y. T. Lee, and H. Cho, *A survey on knowledge transfer for manufacturing data analytics*, Computers in Industry, 104 (2019), pp. 116–130.

[4] J. Blue, D. Gleispach, A. Roussy, and P. Scheibelhofer, *Tool condition diagnosis with a recipe-independent hierarchical monitoring scheme*, IEEE Transactions on Semiconductor Manufacturing, 26 (2012), pp. 82–91.

[5] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola, *Integrating structured biological data by kernel maximum mean discrepancy*, Bioinformatics, 22 (2006), pp. e49–e57.

[6] A. P. Bradley, *The use of the area under the ROC curve in the evaluation of machine learning algorithms*, Pattern Recognition, 30 (1997), pp. 1145–1159.

[7] L. Breiman, *Random forests*, Machine learning, 45 (2001), pp. 5–32.

[8] K. Brinker, *Incorporating diversity in active learning with support vector machines*, in Proceedings of International Conference on Machine Learning, 2003, pp. 59–66.

[9] X. Chai, L. Deng, Q. Yang, and C. X. Ling, *Test-cost sensitive Naive Bayes classification*, in Proceedings of the 4th IEEE International Conference on Data Mining, 2004, pp. 51–58.

[10] L.-C. Chao and L.-I. Tong, *Wafer defect pattern recognition by multi-class support vector machines by using a novel defect cluster index*, Expert Systems with Applications, 36 (2009), pp. 10158–10167.

[11] P. Chen, S. Wu, J. Lin, F. Ko, H. Lo, J. Wang, C. Yu, and M. Liang, *Virtual metrology: A solution for wafer to wafer advanced process control*, in Proceedings of the 2005 IEEE International Symposium on Semiconductor Manufacturing, 2005, pp. 155–157.

[12] C.-F. Chien, K.-H. Chang, and W.-C. Wang, *An empirical study of design-of-experiment data mining for yield-loss diagnosis for semiconductor manufacturing*, Journal of Intelligent Manufacturing, 25 (2014), pp. 961–972.

[13] S. Chopra, R. Hadsell, and Y. LeCun, *Learning a similarity metric discriminatively, with application to face verification*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, IEEE, 2005, pp. 539–546.

[14] M. Desjardins, J. MacGlashan, and K. L. Wagstaff, *Confidence-based feature acquisition to minimize training and test costs*, in Proceedings of the 2010 SIAM International Conference on Data Mining, 2010, pp. 514–524.

[15] A. Dhurandhar and K. Sankaranarayanan, *Improving classification performance through selective instance completion*, Machine Learning, 100 (2015), pp. 425–447.

[16] A. Dogan and D. Birant, *Machine learning and data mining in manufacturing*, Expert Systems with Applications, 166 (2021), p. 114060.

[17] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91 (2002), pp. 201–213.

[18] M. Fan, Q. Wang, and B. van der Waal, *Wafer defect patterns recognition based on OPTICS and multi-label classification*, in Proceedings of IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, 2016, pp. 912–915.

[19] S.-K. S. Fan, S.-C. Lin, and P.-F. Tsai, *Wafer fault detection and key step identification for semiconductor manufacturing using principal component analysis, adaboost and decision tree*, Journal of Industrial and Production Engineering, 33 (2016), pp. 151–168.

[20] L. C. Freeman, *Elementary Applied Statistics: For Students in Behavioral Science*, John Wiley & Sons, 1965.

[21] Y. Gal and Z. Ghahramani, *Dropout as a Bayesian approximation: Representing model uncertainty in deep learning*, in Proceedings of International Conference on Machine Learning, 2016, pp. 1050–1059.

[22] Y. Gal, R. Islam, and Z. Ghahramani, *Deep bayesian active learning with image data*, in Proceedings of International Conference on Machine Learning, 2017, pp. 1183–1192.

[23] Y. Ganin and V. Lempitsky, *Unsupervised domain adaptation by backpropagation*, in Proceedings of the International Conference on Machine Learning, PMLR, 2015, pp. 1180–1189.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT press, 2016.

[25] Q. P. He and J. Wang, *Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes*, IEEE Transactions on Semiconductor Manufacturing, 20 (2007), pp. 345–354.

[26] ——, *Principal component based k-nearest-neighbor rule for semiconductor process fault detection*, in Proceedings of the American Control Conference, IEEE, 2008, pp. 1606–1611.

[27] S. He, G. A. Wang, M. Zhang, and D. F. Cook, *Multivariate process monitoring and fault identification using multiple decision tree classifiers*, International Journal of Production Research, 51 (2013), pp. 3355–3371.

[28] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu, *Batch mode active learning and its application to medical image classification*, in Proceedings of International Conference on Machine Learning, ACM, 2006, pp. 417–424.

[29] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, *Bayesian active learning for classification and preference learning*, arXiv preprint arXiv:1112.5745, (2011).

[30] R. Hwa, *Sample selection for statistical parsing*, Computational Linguistics, 30 (2004), pp. 253–276.

[31] Y.-S. Jeong, S.-J. Kim, and M. K. Jeong, *Automatic identification of defect patterns in semiconductor wafer maps using spatial correlogram and dynamic time warping*, IEEE Transactions on Semiconductor Manufacturing, 21 (2008), pp. 625–637.

[32] S. Ji and L. Carin, *Cost-sensitive feature acquisition and classification*, Pattern Recognition, 40 (2007), pp. 1474–1485.

[33] M. Kampffmeyer, A.-B. Salberg, and R. Jenssen, *Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks*, in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2016, pp. 1–9.

[34] P. Kanani and P. Melville, *Prediction-time active feature-value acquisition for cost-effective customer targeting*, in Proceedings of the Workshop on Cost Sensitive Learning, NIPS, 2008.

[35] P. Kang, D. Kim, and S. Cho, *Semi-supervised support vector regression based on self-training with label uncertainty: An application to virtual metrology in semiconductor manufacturing*, Expert Systems with Applications, 51 (2016), pp. 85–106.

[36] S. Kang, *Joint modeling of classification and regression for improving faulty wafer detection in semiconductor manufacturing*, Journal of Intelligent Manufacturing, 31 (2020), pp. 319–326.

[37] S. Kang, D. An, and J. Rim, *Incorporating virtual metrology into failure prediction*, IEEE Transactions on Semiconductor Manufacturing, 32 (2019), pp. 553–558.

[38] S. Kang, E. Kim, J. Shim, W. Chang, and S. Cho, *Product failure prediction with missing data*, International Journal of Production Research, 56 (2018), pp. 4849–4859.

[39] S. Kang, E. Kim, J. Shim, S. Cho, W. Chang, and J. Kim, *Mining the relationship between production and customer service data for failure analysis of industrial products*, Computers & Industrial Engineering, 106 (2017), pp. 137–146.

[40] 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, 32 (2019), pp. 302–309.

[41] D. P. Kingma and J. Ba, *Adam: a method for stochastic optimization*, in Proceedings of International Conference on Learning Representations, 2015.

[42] A. Kirsch, J. van Amersfoort, and Y. Gal, *BatchBALD: Efficient and diverse batch acquisition for deep Bayesian active learning*, arXiv preprint arXiv:1906.08158, (2019).

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, in Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[44] K. Kyeong and H. Kim, *Classification of mixed-type defect patterns in wafer bin maps using convolutional neural networks*, IEEE Transactions on Semiconductor Manufacturing, 31 (2018), pp. 395–402.

[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.

[46] H. Lee, Y. Kim, and C. O. Kim, *A deep learning model for robust wafer fault monitoring with sensor measurement noise*, IEEE Transactions on Semiconductor Manufacturing, 30 (2016), pp. 23–31.

[47] K. B. Lee, S. Cheon, and C. O. Kim, *A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes*, IEEE Transactions on Semiconductor Manufacturing, 30 (2017), pp. 135–142.

[48] D. D. Lewis and J. Catlett, *Heterogeneous uncertainty sampling for supervised learning*, in Proceedings of International Conference on Machine Learning, 1994, pp. 148–156.

[49] L. Li, K. Ota, and M. Dong, *Deep learning for smart industry: Efficient manufacture inspection system with fog computing*, IEEE Transactions on Industrial Informatics, 14 (2018), pp. 4665–4673.

[50] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, *A structured self-attentive sentence embedding*, in Proceedings of the International Conference on Learning Representations, 2017.

[51] C. X. Ling, Q. Yang, J. Wang, and S. Zhang, *Decision trees with minimal costs*, in Proceedings of the 21st International Conference on Machine Learning, 2004.

[52] C.-W. Liu and C.-F. Chien, *An intelligent system for wafer bin map defect diagnosis: An empirical study for semiconductor manufacturing*, Engineering Applications of Artificial Intelligence, 26 (2013), pp. 1479–1486.

[53] M. Long, Y. Cao, J. Wang, and M. Jordan, *Learning transferable features with deep adaptation networks*, in Proceedings of the International Conference on Machine Learning, PMLR, 2015, pp. 97–105.

[54] S. Mahadevan and S. L. Shah, *Fault detection and diagnosis in process data using one-class support vector machines*, Journal of process control, 19 (2009), pp. 1627–1639.

[55] J. Maiora, B. Ayerdi, and M. Graña, *Random forest active learning for aaa thrombus segmentation in computed tomography angiography images*, Neurocomputing, 126 (2014), pp. 71–77.

[56] P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney, *An expected utility approach to active feature-value acquisition*, in Proceedings of the 5th IEEE International Conference on Data Mining, IEEE, 2005, pp. 1–4.

[57] T. Nakazawa and D. V. Kulkarni, *Wafer map defect pattern classification and image retrieval using convolutional neural network*, IEEE Transactions on Semiconductor Manufacturing, 31 (2018), pp. 309–314.

[58] ——, *Anomaly detection and segmentation for wafer defect patterns using deep convolutional encoder decoder neural network architectures in semiconductor manufacturing*, IEEE Transactions on Semiconductor Manufacturing, 32 (2019), pp. 250–256.

[59] S. J. Pan and Q. Yang, *A survey on transfer learning*, IEEE Transactions on knowledge and data engineering, 22 (2009), pp. 1345–1359.

[60] Y. Pan, T. Yao, Y. Li, Y. Wang, C.-W. Ngo, and T. Mei, *Transferrable prototypical networks for unsupervised domain adaptation*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2239–2247.

[61] E. L. Park, J. Park, J. Yang, S. Cho, Y.-H. Lee, and H.-S. Park, *Data based segmentation and summarization for sensor data in semiconductor manufacturing*, Expert Systems with Applications, 41 (2014), pp. 2619–2629.

[62] J. Park, I.-H. Kwon, S.-S. Kim, and J.-G. Baek, *Spline regression based feature extraction for semiconductor process fault detection using support vector machine*, Expert Systems with Applications, 38 (2011), pp. 5711–5718.

[63] F. Perez, R. Lebret, and K. Aberer, *Weakly supervised active learning with cluster annotation*, in Proceedings of the Bayesian Deep Learning Workshop, NeurIPS, 2018.

[64] M. Piao, C. H. Jin, J. Y. Lee, and J.-Y. Byun, *Decision tree ensemble-based wafer map failure pattern recognition based on radon transform-based features*, IEEE Transactions on Semiconductor Manufacturing, 31 (2018), pp. 250–257.

[65] F. Provost and P. Domingos, *Tree induction for probability-based ranking*, Machine Learning, 52 (2003), pp. 199–215.

[66] N. Roy and A. McCallum, *Toward optimal active learning through monte carlo estimation of error reduction*, in Proceedings of International Conference on Machine Learning, 2001, pp. 441–448.

[67] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, *Semi-supervised domain adaptation via minimax entropy*, in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8050–8058.

[68] K. Sankaranarayanan and A. Dhurandhar, *Intelligently querying incomplete instances for improving classification performance*, in Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, ACM, 2013, pp. 2169–2178.

[69] M. Saqlain, B. Jargalsaikhan, and J. Y. Lee, *A voting ensemble classifier for wafer map defect patterns identification in semiconductor manufacturing*, IEEE Transactions on Semiconductor Manufacturing, 32 (2019), pp. 171–182.

[70] T. Scheffer, C. Decomain, and S. Wrobel, *Active hidden Markov models for information extraction*, in Proceedings of International Symposium on Intelligent Data Analysis, 2001, pp. 309–318.

[71] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*, vol. 39, Cambridge University Press Cambridge, 2008.

[72] O. Sener and S. Savarese, *Active learning for convolutional neural networks: A core-set approach*, in Proceedings of International Conference on Learning Representations, 2018.

[73] B. Settles, *Active learning literature survey*, tech. rep., University of Wisconsin-Madison Department of Computer Sciences, 2009.

[74] B. Settles and M. Craven, *An analysis of active learning strategies for sequence labeling tasks*, in Proceedings of Conference on Empirical Methods in Natural Language Processing, 2008, pp. 1070–1079.

[75] H. S. Seung, M. Opper, and H. Sompolinsky, *Query by committee*, in Proceedings of Annual Workshop on Computational Learning Theory, 1992, pp. 287–294.

[76] J. Shim, S. Kang, and S. Cho, *Active learning of convolutional neural network for cost-effective wafer map pattern classification*, IEEE Transactions on Semiconductor Manufacturing, 33 (2020), pp. 258–266.

[77] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, in Proceedings of International Conference on Learning Representations, 2015.

[78] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, *Machine learning for predictive maintenance: A multiple classifier approach*, IEEE Transactions on Industrial Informatics, 11 (2014), pp. 812–820.

[79] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, *Simultaneous deep transfer across domains and tasks*, in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4068–4076.

[80] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, *Deep domain confusion: Maximizing for domain invariance*, arXiv preprint arXiv:1412.3474, (2014).

[81] G. Verdier and A. Ferreira, *Adaptive mahalanobis distance and k-nearest neighbor rule for fault detection in semiconductor manufacturing*, IEEE Transactions on Semiconductor Manufacturing, 24 (2010), pp. 59–68.

[82] C.-H. Wang, *Separation of composite defect patterns on wafer bin map using support vector clustering*, Expert Systems with Applications, 36 (2009), pp. 2554–2561.

[83] J. Wang, Z. Yang, J. Zhang, Q. Zhang, and W.-T. K. Chien, *AdaBalGAN: An improved generative adversarial network with imbalanced learning for wafer defective pattern recognition*, IEEE Transactions on Semiconductor Manufacturing, 32 (2019), pp. 310–319.

[84] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, *Cost-effective active learning for deep image classification*, IEEE Transactions on Circuits and Systems for Video Technology, 27 (2016), pp. 2591–2600.

[85] M. Wang and W. Deng, *Deep visual domain adaptation: A survey*, Neurocomputing, 312 (2018), pp. 135–153.

[86] 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, 28 (2014), pp. 1–12.

[87] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang, *Adversarial domain adaptation with domain mixup*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 6502–6509.

[88] Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang, *Representative sampling for text classification using support vector machines*, in Proceedings of European Conference on Information Retrieval, Springer, 2003, pp. 393–407.

[89] Y. Yang, Z. Ma, F. Nie, X. Chang, and A. G. Hauptmann, *Multi-class active learning by uncertainty sampling with diversity maximization*, International Journal of Computer Vision, 113 (2015), pp. 113–127.

[90] D. You, X. Gao, and S. Katayama, *Multisensor fusion system for monitoring high-power disk laser welding using support vector machine*, IEEE Transactions on Industrial Informatics, 10 (2014), pp. 1285–1295.

[91] J. Yu and X. Lu, *Wafer map defect detection and recognition using joint local and nonlocal linear discriminant analysis*, IEEE Transactions on Semiconductor Manufacturing, 29 (2015), pp. 33–43.

[92] Z. Zheng and B. Padmanabhan, *On active learning for data acquisition*, in Proceedings of the 2nd IEEE International Conference on Data Mining, IEEE, 2002, pp. 562–569.

[93] Z. Zhou, C. Wen, and C. Yang, *Fault detection using random projections and k-nearest neighbor rule for semiconductor manufacturing processes*, IEEE Transactions on Semiconductor Manufacturing, 28 (2014), pp. 70–79.

국문초록

예측 모델링은 지도 학습의 일종으로, 학습 데이터를 통해 입력 변수와 출력 변수 간의 함수적 관계를 찾는 과정이다. 이런 예측 모델링은 육안 검사 자동화, 불량 제품 사전 탐지, 고비용 검사 결과 추정 등 제조 시스템 전반에 걸쳐 활용된다. 높은 성능의 예측 모델을 달성하기 위해서는 양질의 데이터가 필수적이다. 하지만 제조 시스템에서 원하는 종류의 데이터를 원하는 만큼 획득하는 것은 현실적으로 거의 불가능하다. 데이터 획득의 어려움은 크게 세가지 원인에 의해 발생한다. 첫번째로, 라벨링이 된 데이터는 항상 비용을 수반한다는 점이다. 많은 문제에서, 라벨링은 숙련된 엔지니어에 의해 수행되어야 하고, 이는 큰 비용을 발생시킨다. 두번째로, 검사 비용 때문에 모든 검사가 모든 제품에 대해 수행될 수 없다. 제조 시스템에는 시간적, 금전적 제약이 존재하기 때문에, 원하는 모든 검사 결과값을 획득하는 것이 어렵다. 세번째로, 제조 환경의 변화가 데이터 획득을 어렵게 만든다. 제조 환경의 변화는 생성되는 데이터의 분포를 변형시켜, 일관성 있는 데이터를 충분히 획득하지 못하게 한다. 이로 인해 적은 양의 데이터만으로 모델을 재학습시켜야 하는 상황이 빈번하게 발생한다. 본 논문에서는 이런 데이터 획득의 어려움을 극복하기 위해 능동 학습, 능동 피쳐값 획득, 도메인 적응 방법을 활용한다. 먼저, 웨이퍼 맵 패턴 분류 문제의 높은 라벨링 비용을 해결하기 위해 능동학습 프레임워크를 제안한다. 이를 통해 적은 라벨링 비용으로 높은 성능의 분류 모델을 구축할 수 있다. 나아가, 군집 단위의 라벨링 방법을 능동학습에 접목하여 비용 효율성을 한차례 더 개선한다. 제품 불량 예측에 활용되는 검사 비용 문제를 해결하기 위해서는 능동 검사 방법을 제안한다. 제안하는 새로운 불확실성 추정 방법을 통해 고비용 검사 대상 제품을 선택함으로써 적은 검사 비용으로 높은 성능을 얻을 수 있다. 반도체 제조의 웨이퍼 불량 예측에서 빈번하게 발생하는 레시피 변경 문제를 해결하기

위해서는 도메인 적응 방법을 활용한다. 비교사 도메인 적응과 반교사 도메인 적응의 순차적인 적용을 통해 레시피 변경에 의한 성능 저하를 최소화한다. 본 논문에서는 실제 데이터에 대한 실험을 통해 제안된 방법론들이 제조시스템의 데이터 획득 문제를 극복하고 예측 모델의 성능을 높일 수 있음을 확인하였다.

주요어: 예측 모델링, 데이터 획득, 제조 시스템, 능동학습, 능동적 피쳐값 획득, 도메인 적응, 웨이퍼맵 패턴 분류, 불량 예측, 불량 탐지 및 분류 모델

학번: 2018-39406