The Report committee for Ranjan Subbaraya Radhamohan
Certifies that this is the approved version of the following report:

Automatic Semiconductor Wafer Map Defect Signature Detection Using a Neural Network Classifier

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor: ______________________________
Joydeep Ghosh

____
Mohamed El-Hamdi

Automatic Semiconductor Wafer Map Defect Signature Detection Using a Neural Network Classifier

by

Ranjan Subbaraya Radhamohan, B.S.E.E.

REPORT

Presented to the Faculty of the Graduate School
of The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN
DECEMBER 2010

Automatic Semiconductor Wafer Map Defect Signature Detection Using a Neural Network Classifier

by

Ranjan Subbaraya Radhamohan, M.S.E.
The University of Texas at Austin, 2010

Supervisor: Joydeep Ghosh

The application of popular image processing and classification algorithms, including agglomerative clustering and neural networks, is explored for the purpose of grouping semiconductor wafer defect map patterns. Challenges such as overlapping pattern separation, wafer rotation, and false data removal are examined and solutions proposed. After grouping, wafer processing history is used to automatically determine the most likely source of the issue. Results are provided that indicate these methods hold promise for wafer analysis applications.

Table of Contents

Chapter 1

Introduction

1.1 The Problem of Cost

The chips behind today's electronic devices come from a relatively select group of world-class manufacturers with the capital necessary to build advanced semiconductor fabrication facilities. Samsung, Intel, Globalfoundries, Hynix, Micron, Freescale, and more all compete in a worldwide market with an estimated size of 250 billion dollars [1]. Many of the designs they produce are developed in-house, but there are an increasing number of companies who produce intellectual property only and contract out their designs for manufacturing. ARM is one of the most successful of these companies – its designs can be found in millions of mobile devices worldwide, but it doesn't own a single plant of its own.

The reason for the gradual consolidation in manufacturing and emergence of design houses is cost. State-of-the-art semiconductor fabrication facilities, called *fabs* for short, represent massive investments on the part of their owners. A modern plant with the capacity to produce fifty million chips per month can cost upwards of four billion dollars in initial investment, plus hundreds of millions in maintenance over its lifetime [2]. Not only are costs high, but there is a relatively limited window in which to earn a return on that investment. At the rate of technological advances, a plant may become obsolete and uncompetitive in only eight to ten years [3].

There are several ways to maximize return on investment. One is to increase the price of the product, which is difficult as many chips, such as the memory chips used in USB drives and MP3 players, have become commodities and earn profits measured in tens of cents [4]. Another is to fabricate as many chips as possible during the plant's lifetime, which major manufacturers unquestionably strive to do. Fabs run twenty four hours a day, three hundred and sixty five days a year, and may produce billions of chips over their lifetime. A third, less visible method is to increase the percentage of chips that pass final electrical tests and can actually be sold. This number, called *yield*, can dramatically affect the long term output and money making potential of a plant.

1.2 Wafer Yield

All modern chips start production as part of a circular sheet of silicon called a wafer. What follows is a complex recipe of hundreds or thousands of steps that may take thirty to ninety days from start to finish. There are three core operations that are repeated throughout the process. The first is to grow or deposit a film of polycrystalline silicon, silicon oxide, or metal thousands of times thinner than the diameter of a human hair. The second is to create a pattern mask on top of the film using photo-lithography techniques. The effect of this mask is similar to that of a stencil – some areas of the film are exposed while others are covered. The third is to use a chemical etch to remove only areas of the film that are left exposed by the mask. After etch, the mask is removed and a patterned film remains. This process repeats tens of times per wafer, with each new film sitting on top of the film that came before it. These layers interconnect electrically, and a chip is thus created from silicon upwards, like a new skyscraper rising floor by floor.

A single wafer may have hundreds or thousands of chips arranged in a tight grid on its surface. After production is complete, each one of these chips is physically and electrically stressed to induce failure in those that are weak. The percentage of chips on each wafer that pass these tests and can be sold is the wafer's yield. Since the cost to produce each wafer is largely fixed, increasing the number of chips that can be sold from each results directly in increased profits.

1.3 Far from 100%

Despite the sophistication and expense of modern semiconductor manufacturing tools, achieving 100% yield on a wafer for any type of chip is a difficult and rare accomplishment. All processes in the fab have natural variation, and though it may only be measured on a nanometer scale, it can have a significant effect on the electrical operation of circuit features that are themselves only tens of nanometers in width. Variation occurs not only from wafer to wafer processed on a tool, but within a single wafer itself, often from center to edge radially. The layering of hundreds of varying center-to-edge thickness and circuit dimension profiles over the course of wafer production can lead to areas of the wafer having significantly weaker electrical performance than others.

On top of process variation, all manufacturing equipment has the potential to generate specific defects on the wafer surface. Due to physical movement of tool components, chamber pressurization systems, wafer handling surfaces, and process imperfections, each tool may deposit a handful to several thousand micron-scale particles on the wafer surface. Depending on what step of the process these defects fall at, their effects can be devastating to the operation of the chip. A single particle can break an electrical connection or cause a structure to be severely malformed, rendering the chip inoperable when tested.

Other defects are caused by the inadequate removal of sacrificial films. For example, during the photo-lithography process, an organic photo-sensitive material called photo-resist is spun onto the wafer surface. After the material has served its purpose, it needs to be removed. This is usually done with a dual step process that combines a plasma ash with a chemical wash. Unfortunately, depending on the thickness of the photo-resist, this removal process is not always completely effective, sometimes leaving behind large, thick, crusty residue that acts as an electrical insulator and can prevent chip operation.

Other defects can include physical damage to the wafer itself. Chemical mechanical polish (CMP) tools, which are used to improve the within-wafer thickness uniformity of films before patterning and etching, have diamond-studded pads that polish the wafer surface with high precision. Even the slightest imperfection on the pad can leave hundreds of microscopic gouges in the top film and those below it, cutting electrical lines and preventing the proper patterning of subsequent layers.

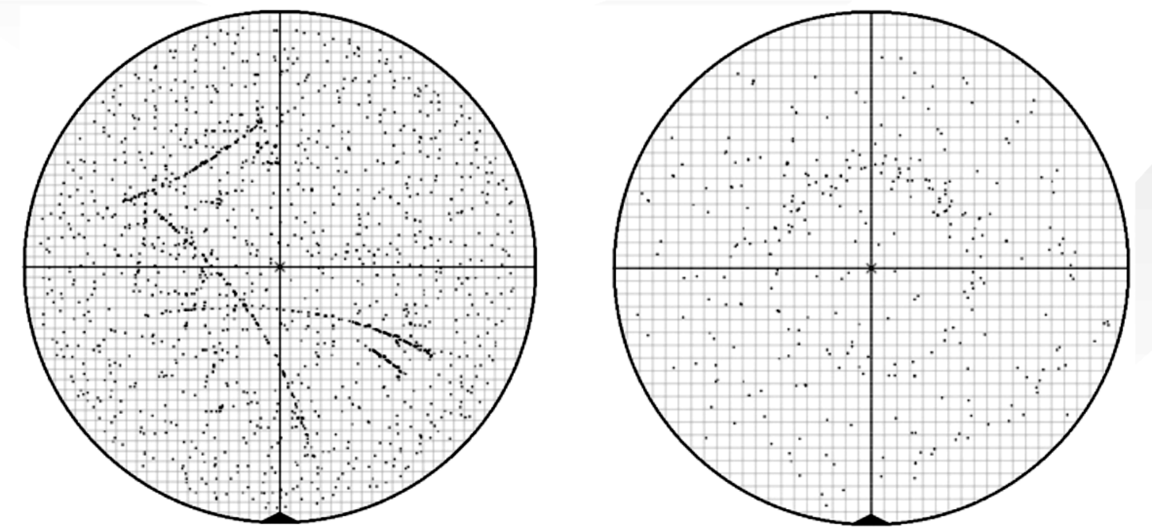


Figure 1: Two real-world examples of defect maps. On the left, a map exhibiting multiple CMP scratch signatures. On the right, a light ring signature near center [5].

Additionally, almost every tool in the fab has robotics designed for wafer handling – moving the wafer from step-to-step within the tool. If mis-calibrated or defective, the robots can leave large scratches on the wafer that are visible to the naked eye. These scratches not only destroy all chips they cover, but can later be a source of particle defects, prevent photo-lithography tools from properly focusing, or create a weakness in the wafer's structure that causes it to shatter in later high temperature furnace processes.

1.4 Defect and Signature Detection

Fortunately, tools exist to help detect defects while the wafer is still in production. Equipment manufacturers such as KLA Tencor, Hitachi, and Applied Materials all supply advanced wafer surface scan tools to semiconductor fabs. These tools operate by scanning over the wafer surface and comparing each chip to its neighbors using image comparison algorithms. A difference that is present between the current chip and both of its neighbors is flagged as a defect by the tool and its location and approximate size are recorded. The output of a complete scan, a list of all defects found on the wafer, can be displayed by defect analysis software as a spatial point map.

Defect point maps are obtained at numerous steps throughout the line. Each major group of the three core semiconductor processes can have half a dozen or more associated scan steps. Depending on the tool's underlying technology and the sensitivity settings specified by the scan's parameter set, obtaining a defect map can take as little as five minutes or as much as one to two hours. At a major fab, this allows thousands of wafers to be sampled and inspected daily. Still, the percentage of wafer-step opportunities actually scanned is usually under ten percent due to capacity constraints. It is therefore critical to make the most of the data that is obtained.

1.5 Making Use of the Maps

Defect maps are invaluable for semiconductor yield enhancement engineers. Not only are the raw statistical measures they provide essential for monitoring tool and process health, but the spatial clusters that emerge can act as fingerprints for specific issues. These clusters, called *signatures*, can help identify what process within a tool caused the defect issue. For example, a swirl signature may point to a problem with a center chemical dispense mechanism that operates while the wafer is rotating. A large, horizontal scratch on the wafer may match exactly with the path of a robot's arm, suggesting the presence of a burr or other imperfection on its surface. A pattern that repeats at even intervals across the wafer surface can indicate a problem with the photo-lithography tool, which exposes the photo-resist shot by shot as it moves in a serpentine motion.

An interesting consideration when analyzing wafer maps is that a signature on a particular scan is not always from the process step that immediately precedes the scan. Instead, the problem may have occurred several steps prior to detection. Since then, multiple processes may have taken place, causing the signature to evolve visually into something that is not always recognizable. For example, a deep scratch in the wafer may look like a solid line of defects if scanned directly after it occurred. After several steps of processing, however, the particles generated by the initial scratch may have streamed across the wafer, significantly enlarging the affected area and changing the the shape and density of the defect distribution. To a human observer, the original source of the issue may be easily discernible. To a computer algorithm, however, the signature may be considered separate from the original scratch.

Another challenge is how to interpret overlapping signatures. By the time a wafer is sampled and scanned, it may have accumulated several distinct signatures at various steps. To accurately diagnose its various issues, these signatures must be considered separately, even though they overlap spatially. Again, while an engineer would have little difficulty separating standard signatures from one another, computer algorithms struggle.

1.6 Commonalities

When signatures like those mentioned above show up on multiple wafers over time, it becomes possible to perform what is called a *commonality*. A commonality finds which tool or process chamber all of the affected wafers ran on, and over what time range. If a particular tool has run all of the wafers with a certain signature, and it has done so over a relatively short period of time, it is a likely suspect as the source of the issue.

Adding to their strength, commonality tools also include equipment run percentage information for each step. If the percentage of material processed by a suspect tool at a step is high, then the commonality is weaker, because there is more chance that the affected wafers could have all run on that tool by coincidence. If the run percentage is low, the fact that the suspect tool still produced all of the affected material makes it more suspicious.

Thanks to their speed, simplicity, and overall effectiveness, wafer map commonalities are one the most important tools available to a semiconductor yield engineer. In practice, finding a commonality for a signature is far more important than attempting to classify the signature as a certain type such as ring or scratch. This is because it allows the offending tool to be shut down immediately, thus preventing further material from being affected. After the incident is contained, investigation can commence as to what is wrong with the equipment and what should be done with affected material.

Chapter 2

The Problem and a Proposed Solution

One of the most striking aspects of semiconductor manufacturing is the sheer volume of data that is generated. Data collection is not limited to the defect scan tools described in the introduction. Vast quantities of information are reported by the process tools themselves, from millisecond resolution parameter traces for items such chamber pressure and RF power, to on-board metrology for intermediate steps in the process. It is almost inevitable that a large percentage of this data is wasted – not covered under automatic statistical monitoring, and certainly not manually monitored and analyzed by engineers or technicians. Any systems that can be designed to automatically search for signals in this sea of data can be of great benefit, providing visibility to issues that humans simply do not have the time to search for.

2.1 Wafer Map Analysis Today

Wafer map signature analysis today is a process largely based on manual map review and wafer-level statistical trend monitoring. Because of its importance, human engineers review a portion of the maps in what time they can spare, looking for relatively gross failures and clear repetition from wafer to wafer. However, not all of the thousands of wafer scan maps produced a day can be reviewed, and more subtle spatial signatures are easily missed. Also, signatures that appear only intermittently are unlikely to be recognized by an engineer who looks through thousands of maps a week, or by a team of engineers who alternate monitoring days.

Automated statistical process control tools monitor defect count trends and can flag deviations, but have no visibility to spatial variations. For example, the total defect counts for a scan may remain constant, but the wafer map signature density may have increased in a particular area of the wafer and been offset by a reduction in another area. Limited regional trend monitoring is sometimes done, dividing the wafer into concentric rings for center, middle, and edge regions. However, this is still relatively crude. In practice, total defect count for a wafer is by far the most commonly used metric. Statistical measures serve more to highlight a problem once it has crossed a certain threshold than to give visibility the phases leading up to an actual issue. Unfortunately, wafers failing these statistical measures may be the *only* maps actually reviewed by an engineer or technician, given time and manpower constraints.

After signature identification, generating affected wafer lists to use when running commonalities is also a manual process. Engineers must look through thousands of historical maps for a similar signature, create a list of material that shows the issue, and enter the list into a commonality tool. Once the list is generated, it must be maintained as new scan data becomes available, and the commonality re-run every time. This is a time-consuming process that limits the number of signatures that can be analyzed and to what depth the problem can be understood.

2.2 A Software Solution for Wafer Map Analysis

Large quantities of data and limited resources call for an effective, automated solution for spatial signature grouping and commonalities. A tool that can handle scan data from various products, tools, and steps, group together similar spatial signatures, and then automatically run commonalities on the groups to find possible sources in the fab would be extremely valuable.

This paper proposes a new software strategy for automatic wafer map analysis. The proposal combines several popular data analysis and manipulation techniques that are often seen in other applications such as image and character recognition. The selected algorithms were tested using representative wafer map data, and showed promise for this application. Future work includes integrating these algorithms into a full-featured wafer map analysis application for production use.

The wafer map analysis software proposed here accepts raw defect data from thousands of scans a day. Once imported, data preparation involves compression and outlier data removal. Also at this stage each map is split into several submaps, each containing a subset of the defect data for a particular defect size range.

Next, the program takes the submaps and separates each signature present. The separation is accomplished using agglomerative spatial clustering techniques. Secondary information such as defect density is used to aid separation. This signature separation splits the original submaps into further submaps.

Finally, the submaps are converted into binary maps. Instead of maintaining defect counts across the signature, locations with a signature are given a value of 1, and locations without a signature are given a value of 0. This helps reduce the effect of signal strength variation from map to map.

After data preparation, the signature submaps are run through a classifier to group similar issues. For this proposal, a multilayer neural network was used. The classifier attempts to match the signature with its known library of signatures. If there is a match, the containing map is added to that signature's group. If there is no close match, the signature is added to the library as a new class. In both cases, the classifier is trained with the new information.

After classification, commonalities are run on the known signature groups in the library. The commonality shows what process tools and steps all members of each group share in their histories. These common tools are the most likely sources of the signature.

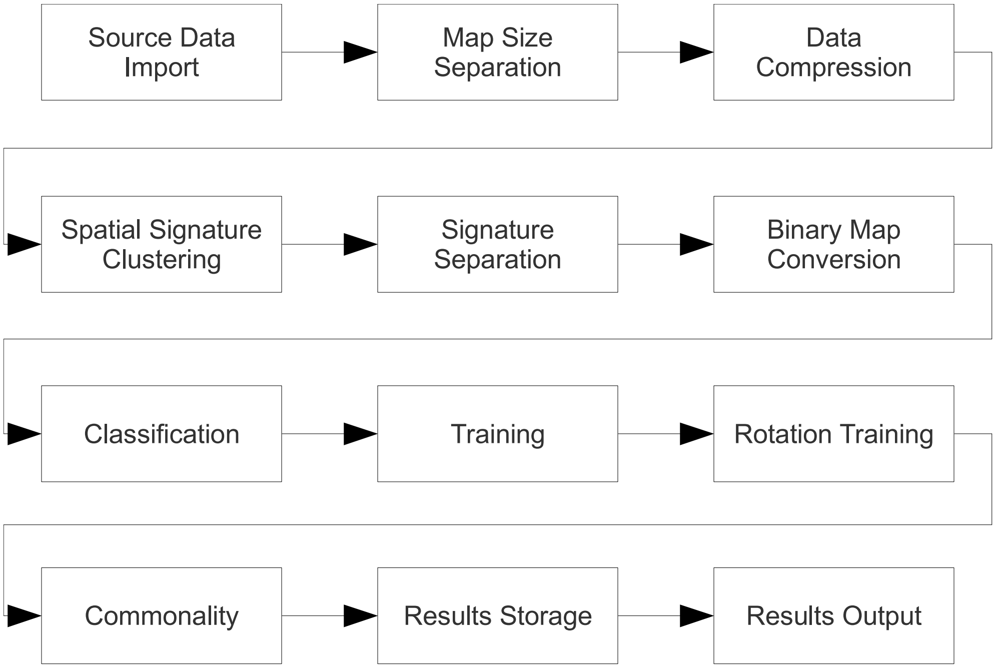


Figure 2: The proposed automatic wafer map classification software flow.

After the algorithm is complete, results are presented to the engineer that show the various signatures found and the process equipment that is suspected to have created them. The engineer can use this data to begin investigation of a particular issue or immediately take action against a troublesome tool. New data is added to the classifier daily, and groups become more and more defined as the map library is built up over time.

Chapter 3

Related Research for this Problem

As an important problem in the semiconductor manufacturing industry, several research studies have already been conducted in an attempt to provide automatic detection and classification of wafer map signatures. A portion of the related research attempts to classify signatures by their general type – scratch, spiral, ring, and others. Other work leans towards general signature groupings. Undoubtedly, there is also much work being done within the walls of today's leading semiconductor manufacturers that is not published.

One piece of work, "A Rule-Based Recognition of Spatial Defect Patterns on Semiconductor Wafers," [6] presents a solution for signature classification into well known types. In this method, wafer map signatures are first separated into two categories - global and local. Signatures that cover a large portion of the wafer surface, such as rings and spirals, are considered global. Signatures that only affect a smaller part of the wafer surface, like lines and scratches, are considered local. The method first examines all global signatures from each wafer. The map, with global signatures only, is divided into concentric tracks with radially cut sectors. Next, a set of criteria is applied to the sectors that is consistent with known global signatures. For example, a circle signature typically has similar defect densities across all sectors in a track. If a set of tracks on any given wafer passes this criteria, the algorithm classifies the wafer as having a circle signature.

For signatures that are considered local, the algorithm performs spatial clustering. Once the clustering is complete, rules can be applied to the clusters to determine what signature they represent. Cluster properties such as elongation, length, angle, and linearity are all used. For example, if elongation, linearity, and length pass a predetermined criteria, a rule tells the algorithm to check the angle of the cluster with respect to some fixed reference point. If the angle is close to zero or 90 degrees, it is classified as a line. Clusters at other angles are classified as standard scratches.

This type of rule-based method is elegant in its simplicity. The actions of the algorithm are very clear, and with a strong set of rules for each signature of interest, classification can be very reliable. Still, the method requires rules for each signature type to be created in advance. These rules need to general enough to cover signature variation from wafer-to-wafer, but specific enough to still separate unrelated signatures. Depending on the number of rule sets created, there may not be enough distinct signature groups to allow meaningful commonalities to be run.

A second paper, “Automatic Classification of Spatial Signatures on Semiconductor Wafermaps” [7], describes the Spatial Signature Analysis (SSA) algorithm. SSA assumes that all signatures fall into one of four main classes - global, curvilinear, amorphous, and micro-structure. A defect map is converted into an image where a grayscale value is applied to each pixel based on the number of defects in the chip it represents. A spatial clustering algorithm is then used to separate signatures. These signatures are placed into one of the four main groups, and classification begins. Each of the four groups has different key features – for example, curvilinear signatures have properties such as elongation and compactness. These features are computed for each signature and fed into a “fuzzy” k-nearest neighbors classifier. The “fuzzy” aspect of the classifier allows SSA to assign a probability of belonging to a certain class to each signature. The algorithm showed strong classification performance and included an advanced interface for engineer feedback.

The paper “Spatial Patterns in Sort Wafer Maps and Identifying Fab Tool Commonalities” [8] deserves mention here because of its inclusion of an automatic commonality feature. It does not attempt to perform classification, only clustering for the purposes of the commonality. Clustering is performed on the full wafer data set. Because of the sheer number of chips in this data set, k-means clustering is used as a preliminary, fast method to group similar wafers and remove outliers. After k-means, agglomerative clustering is used on the reduced data set to provide a stronger clustering of the defect maps in chip space. Commonalities are performed on the clusters by analyzing fab run history and time frames of processing, similar to the algorithm in this paper. The application produced by the authors of the paper is notable for its excellent visualizations.

Two final papers are of interest here because they apply neural networks to the wafer map classification problem. The first, "A Neural-Network Approach to Recognize Defect Spatial Pattern in Semiconductor Fabrication" [9], proposes a algorithm called the Adaptive Resonance Theory Network 1 (ART1). A neural network is constructed where the number of die on the wafer defines the number of input nodes. Unlike many image classification neural networks, the method proposed in this paper performs unsupervised learning – no classification results are ever fed back to the algorithm for tuning. Instead, a vigilance test is used to learn new signature patterns without forgetting or overwriting old ones. The vigilance test checks how different the current signature is from previously seen signatures. If the new signature is similar enough to a known signature, the classifier is updated with the new information to make it more robust. If the new signature does not appear related to any known signature, a new classification is created by adding an output neuron to the neural network classifier. This method allows the algorithm to both learn new signatures and reinforce knowledge about existing signatures, all without any classification confirmation.

The second paper, “Defect Spatial Recognition Using a Hybrid SOV-SVM Approach in Semiconductor Manufacturing” [10], presents another neural network approach. Here, features such as entropy, energy, contrast, local homogeneity, mass, centroids, and geometric moments are extracted from the wafer map. Instead of using chip values as inputs to the neural network, these features for each wafer become the inputs. The authors combined a self-organizing map, a type of unsupervised neural network, with a type of supervised neural network called a support vector machine. By doing so they were able gain performance on very large data sets yet still maintain a good level of accuracy. Unlike this report's solution, the algorithm was applied to binary electrical maps, not defect maps.

Chapter 4

Defect Data Preparation

The following four chapters explain the design and experimental results of the proposed automatic wafer map analysis system. Each chapter describes a core component of the system – defect data preparation, signature separation, classification, and the fab commonality. Each component presented its own challenges and design decisions. All initial experimentation and tuning was done using Octave, an open-source numerical computing language. Based on the the results from the Octave test runs, a final set of algorithms was implemented in Java, SQL, and Windows Script. Development and testing was split between a 1.8 GHz Intel Pentium M machine with 1.2 GB of RAM running Ubuntu Linux 10.10 and a 2.0 GHz Intel Core 2 Duo machine with 4 GB of RAM running Mac OS 10.6.5. Where relevant to results, the particular machine used is noted.

4.1 Data Extraction

A challenge when working with scan tool data is the format it is saved in. Due to the prominence of an old industry format, defect lists are not automatically saved into a database or even a well-organized XML file. Instead, they are output to a flat text file nicknamed “klarf” after its extension, KRF, or KLA Results File.

For this report, a Windows script was created that parses through KRF files. Keywords are searched for, and data is extracted based on its predicted position in the file with respect to those keywords. Chip coordinates, within-chip location, and approximate size are all obtained for each individual defect. The script also extracts meta-information from the KRF file, such as wafer identification number and scan date. The script is run daily and can process several thousand maps an hour on a modest machine. Results are stored in a SQL Server database table.

4.2 Size Separation

Defect size information from the KRF file is an important tool for separating signatures on a wafer map. It is common in scan analysis to group defects by size, because different process issues and tools tend to cause different size defects. To help separate signatures later, each wafer map is split into several submaps, each containing a subset of the total defects that fit within a certain size range. All of these maps retain the original parent wafer identification information for use in a later commonality, but are effectively treated as separate wafers for spatial clustering and classification. This splitting must be done before compression, described below, because at that point individual defect size data is lost. Keeping with industry standards, the application splits each map into three submaps – one with defects of less than 0.5 μm in size, another with defects between 0.5 μm and 1.0 μm, and a third with defects greater than 1.0 μm.

4.3 Data Compression

Defect scan data sets are very large. Single wafers can occupy dozens of megabytes in plain text format if defect counts are high, since each individual defect is represented by a row of data in the result file. Tens of millions of rows may be generated daily by a group of scan tools. Although the resolution provided by raw data is excellent, it can lead to long calculations and high data storage requirements. One solution is to store the total count of defects for each size range in a chip. This reduces the amount of data stored from thousands of points per wafer to a small multiple of the number of chips on the wafer, usually in the hundreds. Furthermore, reducing to chip-level resolution acts as a smoothing function so slight variations in a signature from wafer-to-wafer have less effect on the classification results.

One disadvantage to using chip resolution for wafer maps is that doing so effectively limits signature comparisons to within a single product. Each type of product has a different rectangular chip footprint, allowing for different numbers of chips to be fit on each standard-size wafer. Attempting to spatially compare a particular chip coordinate from one product to another results in the comparison of different areas of the wafer surface. Signatures, however, appear on the same part of the wafer regardless of the underlying chip layout.

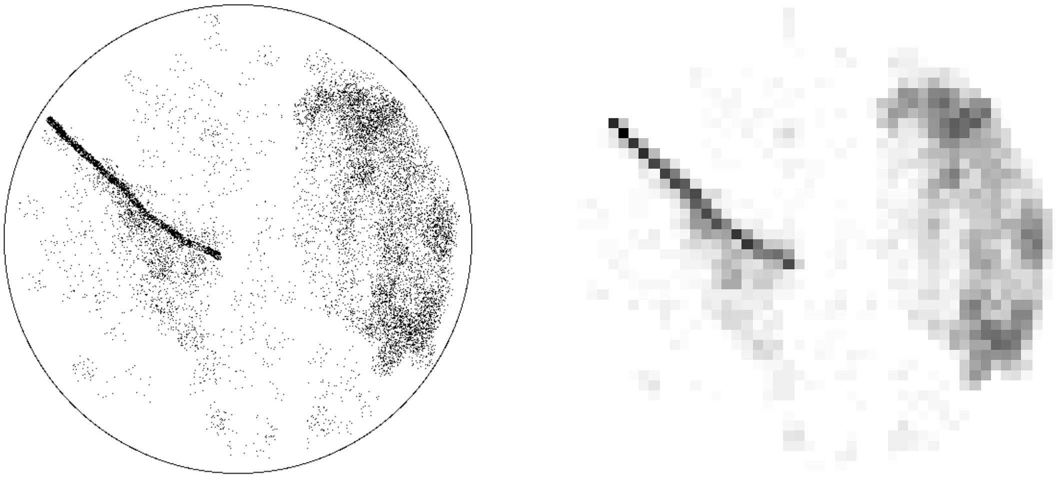


Figure 3: Data compression from a full resolution defect map (left) to a 50x50 general coordinate system (right). The main features of the signatures are preserved.

A standardized coordinate system must be developed to allow signature comparisons across different products. This new coordinate system can be applied to all material by simply translating a defect's product-specific chip coordinate and within-chip location to a general location on the wafer surface.

$$absolute_{x,y} = coordinate_{x,y} \cdot chipsize_{x,y} + withinchip_{x,y}$$

From here, the wafer can be redivided into a standard size grid for compression. A trade-off exists when choosing the resolution for this grid. If it is too low, then the characteristics that separate one signature from another may be lost. Too high a resolution can result in higher sensitivity to noise and variation, plus significantly greater computational requirements for clustering and classification. 50x50 was found to be optimal for this report, and actually approximates chip resolution.

Another consideration is whether to use a standard rectangular coordinate system or a polar one, since the wafer is a circular surface. Assuming evenly spaced concentric subdivisions, the latter results in reduced signature detection resolution at the edges of the wafer where the cells are large. Since a significant percentage of signatures on wafer maps actually occur at the edge of the wafer, due to its proximity to tool components and unfinished extreme edges, this is an undesirable characteristic. A simple rectangular coordinate system can provide equal resolution at center and edge.

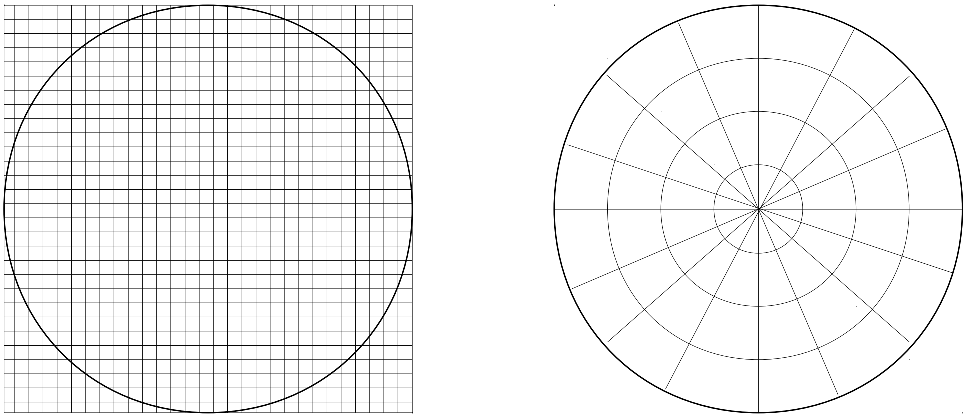


Figure 4: Compression to a rectangular coordinate system provides equal resolution across the entire wafer surface. Radial divisions reduce resolution at the wafer edge.

4.4 Outlier Removal

The scan tools that produce defect maps are susceptible to a variety of software and hardware errors during their operation. To improve classifier performance, maps that are affected by these errors and contain bad data should be removed from the data set. Making this task easier is the fact that when scans fail, the result is usually extreme. Bad scan results may have defect counts two to three orders of magnitude greater than standard material. The scan may not even finish inspecting the entire wafer surface if the defect count passes a set threshold. Real signatures, depending on severity, also may be separated from the overall trend, but rarely to such a high degree.

Removing bad data can be effectively accomplished by determining the normal total count trend for a step and tool, then eliminating points that fall far outside of the distribution. A common method is to utilize the quartiles of the data to determine what qualifies as an outlier [11]. Specifically, the formula below can be used to determine the cutoff point for a one-sided distribution.

$$Cutoff = k \cdot (Quartile_{75} - Quartile_{25})$$

Here, the quartiles represent the lowest 25% and lowest 75% of the data points, and k is a multiplier that can be set based on the nature of the data set. For the defect count trend shown below, $k=6$ is acceptable is remove close outliers, but it may also eliminate real signatures with high defect counts. Much higher values are needed to remove only nonsensical defect scan results. The advantage of the quartile method is simplicity. Other, more computationally-intensive methods can be used, such as k-means, but for this application they are not necessary.

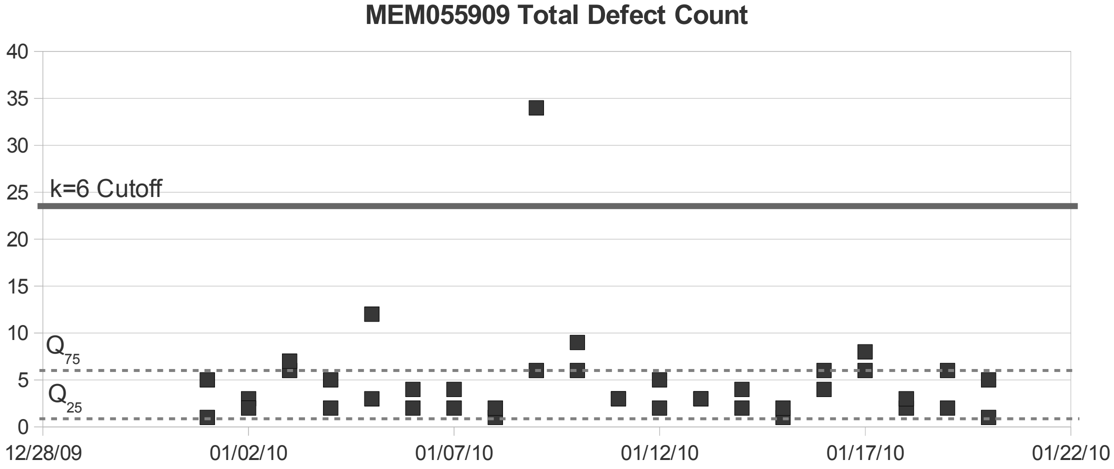


Figure 5: A typical defect count trend with the 1st quartile, 3rd quartile, and computed outlier cutoff shown. The outlier in the trend with 34 defects would be eliminated.

Finally, it is necessary to remove noise from each individual map. This is done in the same manner as above. The first and third quartiles of the defect count data for the map, excluding zeros, are computed. Any spatial units with counts below the third quartile minus the first quartile have their counts set to zero. No *k* multiplier is used in this case, although it could be adjusted if certain products are more sensitive to the presence of defects than others. After this filtering maps are left with only higher defect density spatial units.

Chapter 5

Signature Detection, Separation, and Variation Handling

One of the challenges when working with wafer maps is that multiple signatures often show up on the same scan. These signatures can be from different sources and steps. To successfully group wafers from a single issue, it is therefore necessary to first separate the various signatures on a map. Doing so allows each wafer to potentially belong to multiple signature classes, depending on what signatures are present on its surface.

Signatures also vary from map to map. Each occurrence of a problem may have a slightly different signature. Variation also exists in the measurement itself, between different scan tools or different parameter sets. Other variation can be more extreme, such as signature rotation from wafer to wafer. An automatic wafer map analysis tool needs to be able to handle these forms of variation.

5.1 Spatial Clustering

The separation of signatures on a wafer map is a two dimensional clustering problem. Two popular options were tested – standard K-Means and the computationally-intensive Agglomerative Clustering single-linkage algorithm.

5.1.1 K-Means Clustering

K-means clustering is a simple form of partition clustering. It breaks a set of data points into a predetermined number of clusters through an iterative process. Cluster membership is based on Euclidean distance to moving cluster means.

One of the main weaknesses of k-means clustering is its requirement that the number of final clusters be specified at initialization. For wafer defect maps, it is difficult to estimate how many signatures will be present in advance. Options exist to work around this problem, including the Elbow Method [12]. This method clusters the data set multiple times, each time with a increasing number of clusters. It stops once it finds the number of clusters where a chosen quality measure begins to level off, after which additional clusters do not help explain the data significantly better.

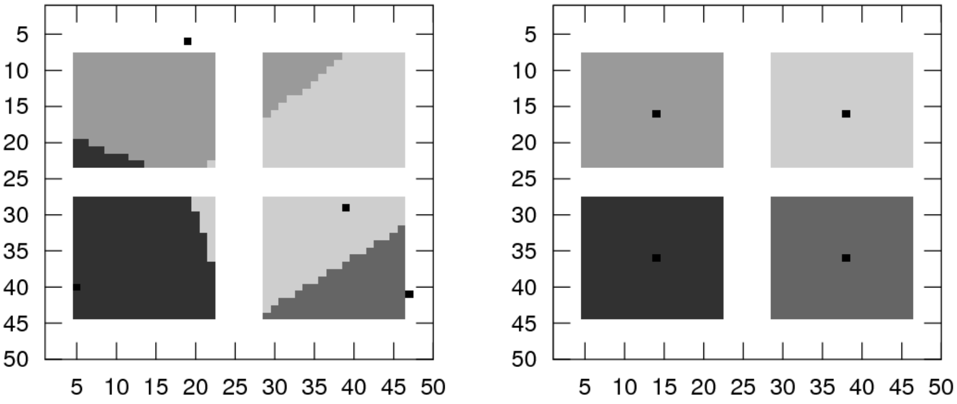


Figure 6: Given the correct number of groups and tight, contained signatures, the K-Means algorithm is effective. The illustration above shows Octave output for the first K-Means iteration (left) and the last (right) of a 50x50 simulated map compression with four signatures. The black dots are cluster means; clusters are different shades of gray.

K-means is inappropriate for wafer map signatures for a more fundamental reason, however. The algorithm is designed for traditional, compact, ball-like clusters [13]. The moving mean eventually finds the centroid of these clusters after a number of iterations. Signatures, however, often are irregular shapes with no self-contained mean. This can cause k-means to split or join signatures incorrectly on a wafer. For example, the wafer below with a large edge ring signature and a smaller center spot signature is impossible to separate using k-means. The algorithm ends up cutting both signatures in half. These false groupings would seriously affect the quality of the classifier and commonality. Maps with both the ring and the spot would be placed in a different class from maps with just the ring or the spot. Commonalities for both the ring and spot would be weakened with the loss of this test data.

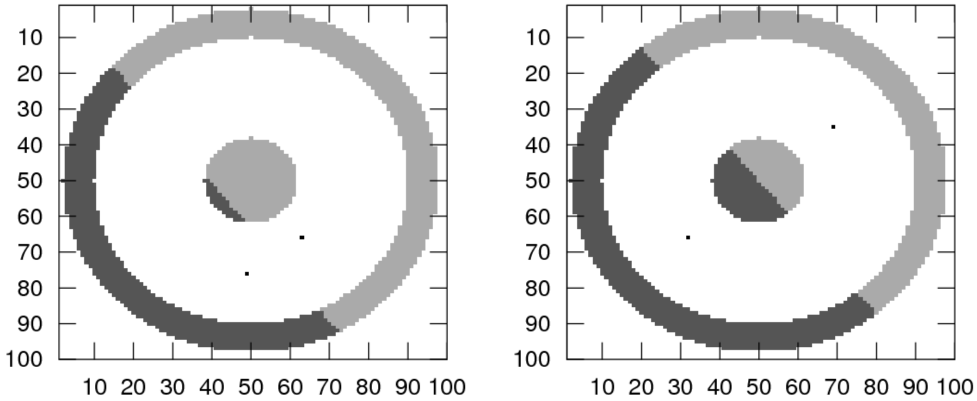


Figure 7: K-Means fails for this simulated 100x100 map compression of two signatures, a center spot and an edge ring. The first iteration is at left, the last at right. Means are shown as black dots, and clusters are differentiated by different gray levels. No matter how the algorithm is initially configured, the signatures will always be split in half.

5.1.2 Agglomerative Clustering

A more appropriate clustering method for spatial wafer map signatures is agglomerative clustering. Agglomerative clustering begins by considering each data point as belonging to its own class. Clusters are then merged if they match certain criteria, usually based on the distance between the two. The algorithm iterates until no more merging occurs, at which point all clusters are formed. This process can be much more computationally intensive than k-means. Clusters, however, can now grow along the irregular path of a signature. The effectiveness of agglomerative clustering is shown in the figure below. The same edge ring and center spot signatures that k-means consistently fails to separate are identified clearly using agglomerative clustering. Note, however, that the map resolution had to be reduced to 50x50 from 100x100 to attain a similar time of calculation.

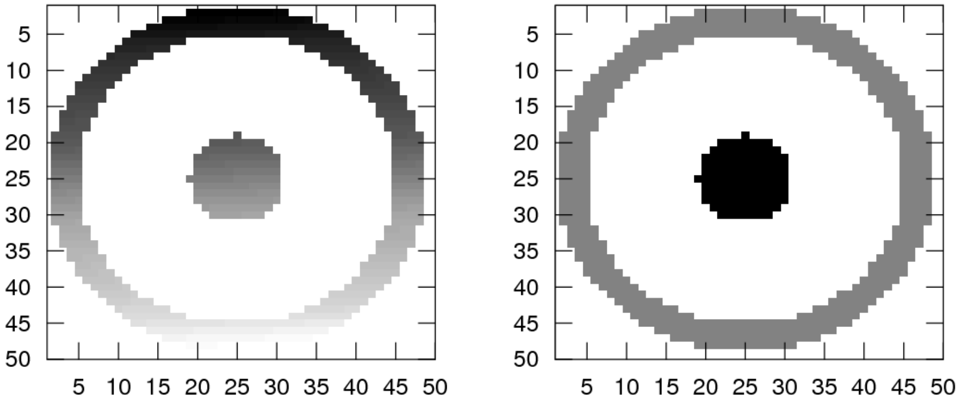


Figure 8: Octave output for an implementation of agglomerative clustering. The left image shows the initial cluster assignments for a 50x50 center spot and edge ring compression. Each shade of gray represents a distinct cluster. After many iterations, the number of classes has reduced to two – one for the ring, and one for the spot.

One of the choices that needs to be made when using agglomerative clustering is what criteria to use when merging clusters. Two common methods are widely used. The first, called single linkage clustering, is to compare the distance of the closest points from the two clusters. The formula below illustrates this distance calculation. Here, CD is the cluster distance, and PD is the Euclidean distance between *x* and *y*, two points in clusters C_X and C_Y, respectively.

$$CD(C_X, C_Y) = \min_{\substack{x \in cluster_X \\ y \in cluster_Y}} PD(x, y)$$

If the cluster distance is below a certain threshold, then the two clusters are completely merged. The second method, called complete linkage, is to compare the distance of the farthest points of the two clusters. As explained in [14], single linkage clustering tends to form long chains as clusters merge with their closest neighbors, while complete linkage clustering better preserves the common concept of a cluster as a tight, spherical group of data points in space. For this reason, in the case of defect clustering, where signatures such as rings and scratches have distinct chain-like properties, single linkage clustering is ideal.

A second consideration with agglomerative clustering is how to distinguish between clusters of interest and noise clusters. The decision was made here to allow small clusters only if they contain high defect count spatial units. Small is defined as covering less than 1% of the wafer area. For a 50x50 map compression, this is all clusters that contain less than 25 spatial units. Of these small clusters, those that do not contain a single spatial unit with a defect count greater than the first quartile of the map's defect count data are discarded. Thus, only high defect density small clusters and large clusters in general are kept for classification.

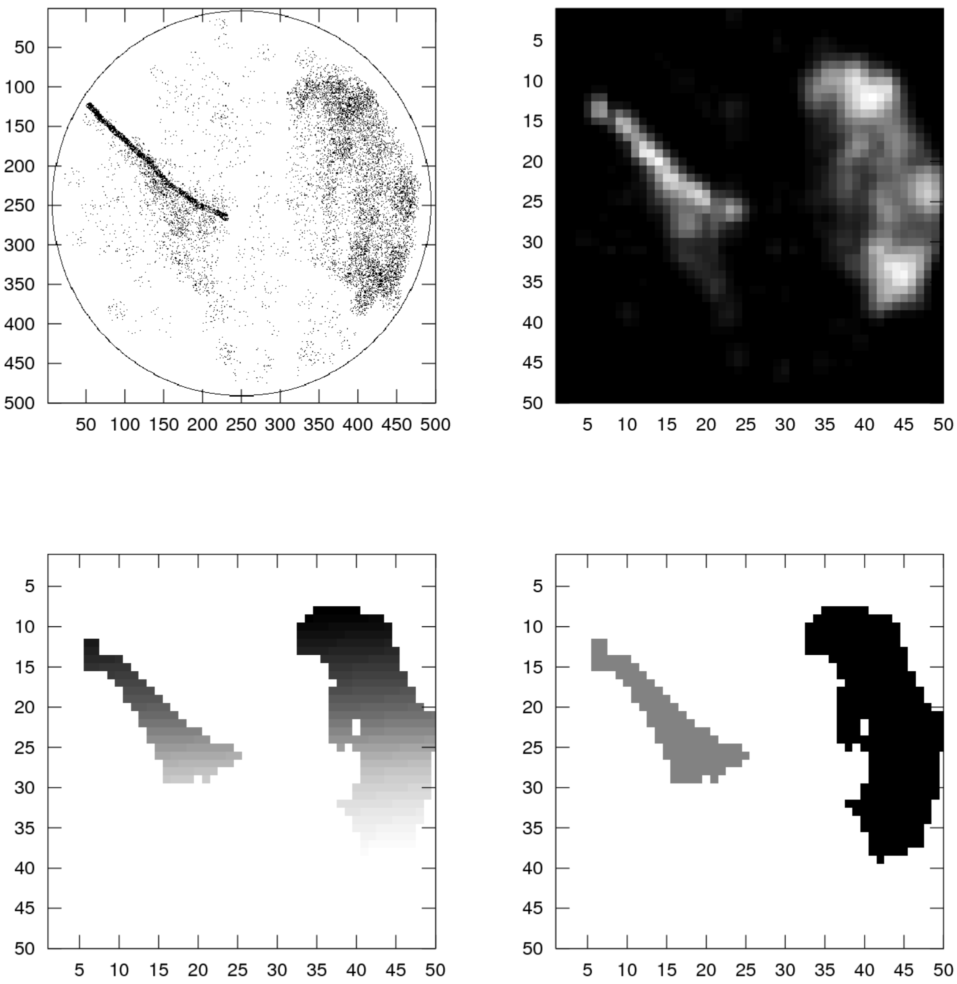


Figure 9: Agglomerative clustering for a realistic map. In the upper left, the original full resolution defect map. In the upper right, its 50x50 compression. Lighter shades of gray indicate higher defect counts. At the lower left, the initial iteration of the agglomerative clustering algorithm where each data point is its own cluster. At the bottom right, the final clustering iteration where the signatures have been separated into two groups.

5.2 Dealing with Signature Overlap

Making the clustering problem more difficult is the reality that there are cases where signatures overlap on a single map. In this case, the chain-based single-linkage clustering method will likely join the overlapping signatures together. A possible solution would be to apply rules to the formed cluster. For example, if it is assumed that signatures are relatively smooth, then a clustered signature with four square edges could be interpreted as the overlap between two signatures, and thus broken in two. However, these assumptions cannot easily be made. The rule in the previous example, for example, fails for a real world "X" signature that is commonly seen from metal deposition tools.

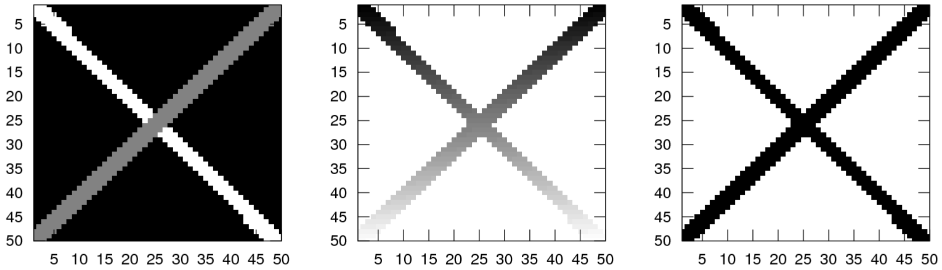


Figure 10: Standard agglomerative clustering takes two distinct bar signatures and merges them into a single X. Defect density information is shown in the leftmost image. The white bar has a higher defect density than the gray bar. This information is wasted with the normal agglomerative clustering algorithm, which only considers distance.

A better assumption is that a signature will have a fairly continuous defect density. Two crossing signatures may have very different densities, especially if one occurred several steps before the other and has been covered by several layers of material. This assumption can actually be incorporated directly into the single-linkage clustering algorithm. In addition to Euclidean distance between the closest points of two clusters, an additional distance measure can be the difference in defect counts between those two spatial units.

$$distance_{spatial} = \min_{\substack{x \in cluster_X \\ y \in cluster_Y}} PD(x, y)$$

$$distance_{density} = |(x_{defectcount} - y_{defectcount})|\left(\frac{pointdistance_{max}}{defectcount_{max}}\right)$$

$$CD(C_X, C_Y) = W \cdot distance_{spatial} + (1 - W) \cdot distance_{density}$$

As shown in the formulas above, the new distance measure, *distance*$_{density}$, is scaled to bring its magnitude in line with the Euclidean measure. The two measures are also inversely weighted with the constant *W*. A larger *W* puts more emphasis on the distance between points, *distance*$_{spatial}$, for clustering. A smaller *W* places more focus on the defect density of the signatures for clustering, so clusters with similar densities can "hop" across spatial boundaries. The choice of *W* is very important to achieving good results for separation. For the 50x50 wafer map compression data used throughout this report, a value of W = 0.45 was found to be optimal for separating signatures of different density but not merging clusters too far apart spatially. Examples of this special agglomerative clustering algorithm are shown in the figures below.

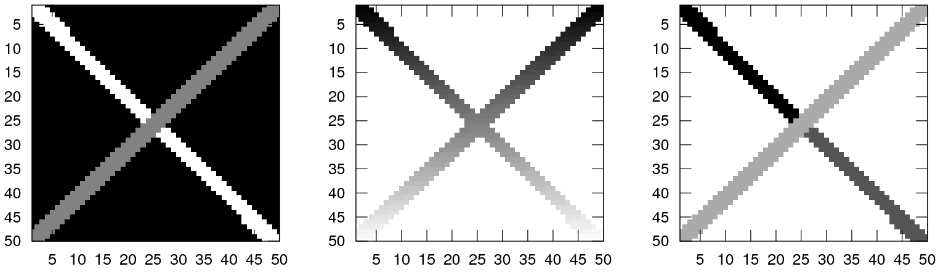


Figure 11: The special agglomerative clustering algorithm in place, but with a poorly chosen value of W. With a high value of W, the algorithm is too heavily weighted towards spatial distance. This splits the white bar into two separate classes due to the overlap of the gray bar. The two clusters cannot "jump" across the boundary to merge.

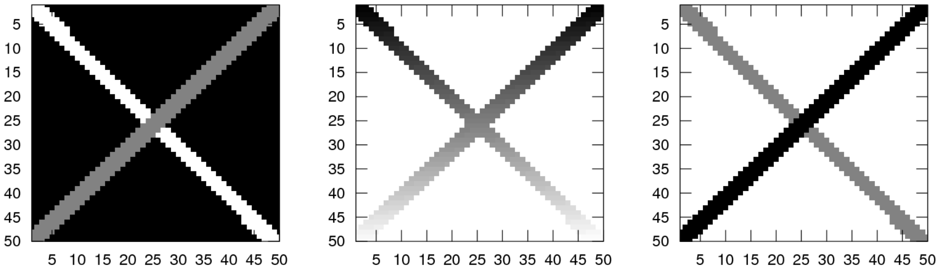


Figure 12: The special agglomerative clustering algorithm with an optimal value of $W = 0.45$. In this case, spatial distance is weighted slightly less than density difference. This allows the two segments of the white bar to hop the physical boundary created by the gray bar and cluster based on their similar defect densities.

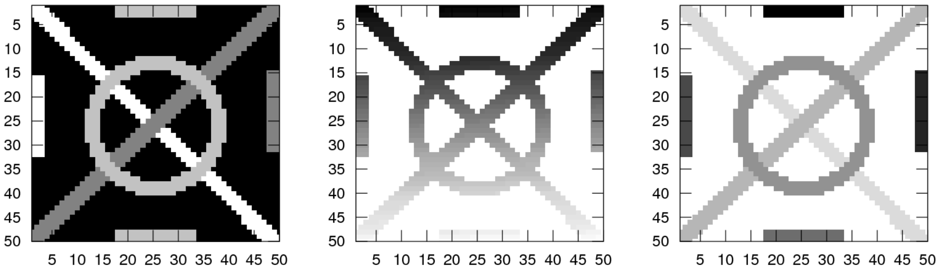


Figure 13: Tested with a challenging signature configuration. Here the algorithm shows its power, as the ring, crossing bars, and edge small bars are all separated correctly.

5.3 Signature Variation from Scan Tools

Scan tools require manual parameter setup at every process step they are used at. The collection of parameters necessary to properly detect real defects and ignore noise is called a *recipe*. Each recipe has a wide range of parameters, from illumination level to angle of incidence to magnification. Depending on the technology used in the tool, different sets of parameters are needed. For example, a tool based on scanning electron microscope imaging will have parameters related to forming the electron beam, while an optical microscope-based tool will have settings related to incident light and polarization.

Recipes are usually created with a randomly sampled wafer at the step of interest. Unfortunately, this wafer may not be sufficiently representative of the majority of wafers at that step. If the parameters are tuned incorrectly, the resulting scan maps can have a significant amount of noise – false defects on the wafer surface. Contributing to the problem is the fact that recipes need to be tuned separately on each inspection tool in the fab, even if the model is identical, because each unit has its own particular tendencies and variations. It is not uncommon for a defect trend to show noticeable signal differences between tools.

To provide clear data to the classifier that avoids as much of this variation as possible, the analysis program converts signature submaps from density maps to binary maps before classification. That is, if a spatial unit has a signature on it, it becomes a 1. If it does not have the signature on it, it becomes a 0. This helps to eliminate signal variation from map to map, and places the focus on shape for classification. Had density also been incorporated in the classifier, scan tool sensitivity differences would have factored more prominently in the results.

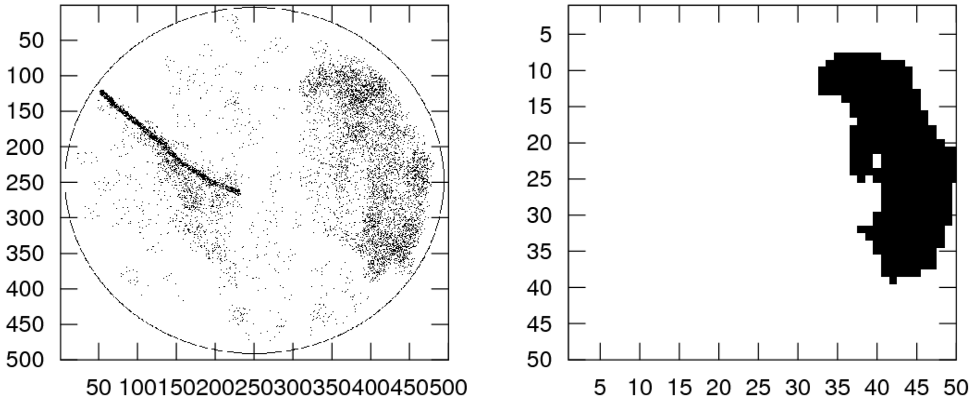


Figure 14: The original full resolution defect map and the resultant separated binary map of the right side patch signature. The signature at right is ready for classification.

5.4 Other Sources of Signature Variation

Many image processing techniques, such as face recognition, are plagued by natural variation from image to image. For example, lighting and shadows, focus, subject angle, and more can all differ slightly from image to image, making classification more difficult. Wafer map signature analysis avoids many of these challenges because of the nature of the data. However, there is still variation from map to map.

Defect density for a signature, for example, typically reduces as scans get farther away from the original occurrence of the problem. Layers build up on top of the signature, obscuring the signal. By clustering by defect density and then converting to a binary map, however, this source of variation is controlled.

Another interesting and frequently seen variation when dealing with signatures is direction. When wafers are processed on tools, they are not always physically aligned to a certain direction. Thus, resulting signatures can appear rotated from map to map when scanned on inspection tools that do align the wafer to the same direction every time. To ready the signature classification algorithm for these rotations, each signature is trained to the classifier multiple times, each at a different degree of rotation. Note that these rotations do not become submaps of the parent wafer map. They are simply used to train to the classifier to make it more robust at handling signature rotation, then dropped.

The rotation algorithm uses trigonometric functions to find the location of each point after rotation by a certain number of degrees *theta* [15]. The coordinate (x_{offset}, y_{offset}) is the point of rotation. For wafer maps, this is chosen as the center of the wafer circle.

$$x_{new} = \cos\left(\frac{\theta \cdot \pi}{180}\right)(x_{old} - x_{offset}) - \sin\left(\frac{\theta \cdot \pi}{180}\right)(y_{old} - y_{offset}) + x_{offset}$$

$$y_{new} = \sin\left(\frac{\theta \cdot \pi}{180}\right)(x_{old} - x_{offset}) + \cos\left(\frac{\theta \cdot \pi}{180}\right)(y_{old} - y_{offset}) + y_{offset}$$

If the algorithm produces a non-integer x_{new} or y_{new}, the new coordinates are simply rounded to the nearest integer boundary. Although in image processing applications some form of interpolation is usually done in this case, it is unnecessary for the relatively crude rotation that is needed here.

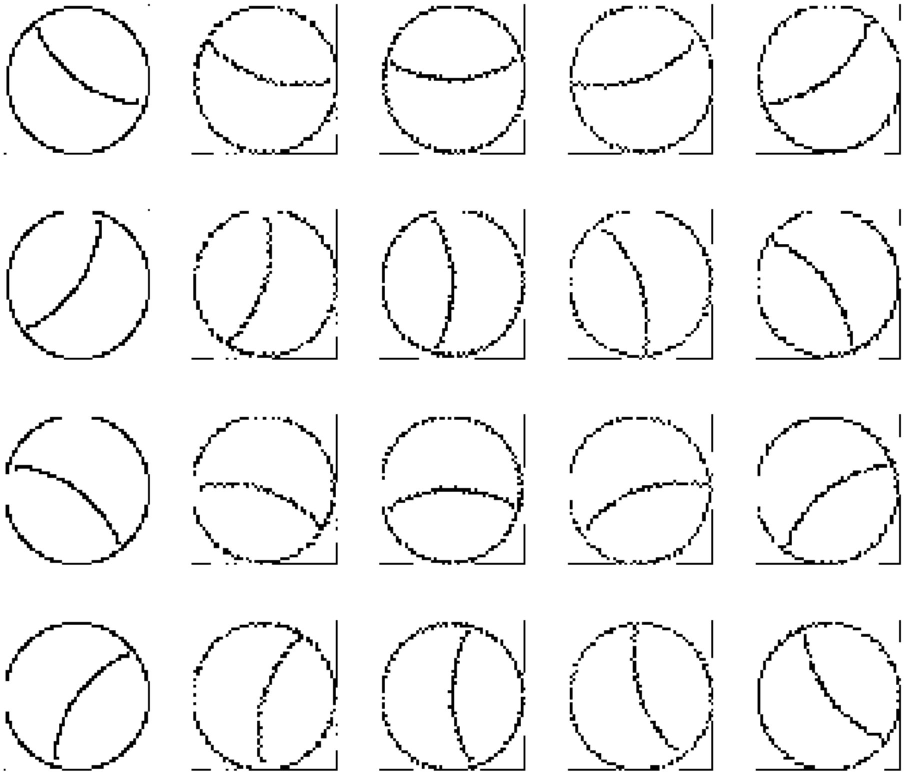


Figure 15: Examples of a map rotated using the algorithm that can be fed into the classifier for training. The algorithm in its current form is relatively crude, and results in some data distortion. Further optimizations are possible but not essential.

Chapter 6

Map Classification and Learning

After data preparation, the algorithm is ready to group signatures. Several methods were attempted in the preparation of this report, yet the method that consistently showed the best performance was a neural network classifier. Neural networks, based loosely on biological processes, have interesting learning abilities that can be used to group patterns such as wafer map signatures.

6.1 Single Layer Neural Networks

In their simplest form, neural networks consist of a single neuron with any number of inputs and a single output. Weight multipliers are applied to each of the inputs to the neuron and the results are summed. A threshold function, $F_{threshold}$, is then used to determine the output of the neuron based on the sum. This output determines the classification of the point. Mathematically, the output value y is determined using the equation below.

$$y = F_{threshold}(x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3)$$

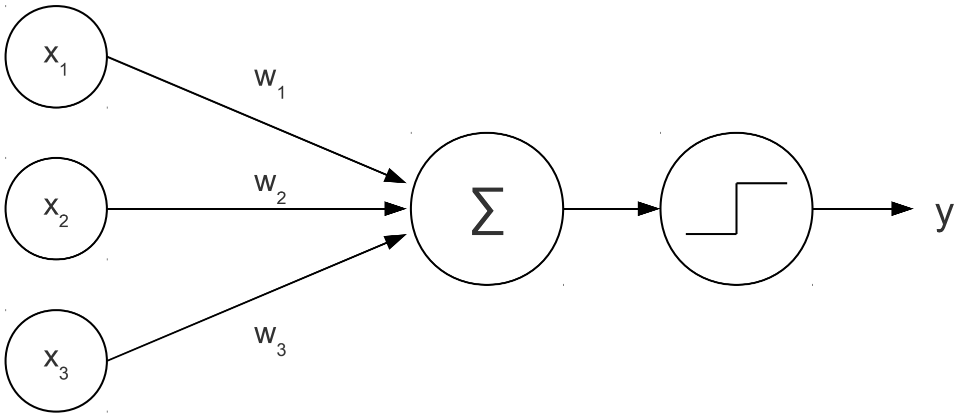


Figure 16: A single neuron with three inputs and a binary threshold function.

The values of the weights are the key to the neural network. They determine how the neuron responds to different combinations of input values. Selecting the correct weights is done iteratively through a supervised process called training.

Training a single neuron classifier involves the comparison of the output produced with the output expected for a given set of input values. Each input value gets multiplied by the difference between the two values and a chosen learning rate. The result is then added to the weight for the input.

$$w_{new} = w_{old} + rate_{learn}(output_{expected} - output_{actual}) \cdot input$$

After training the neuron with multiple samples for each class, the weights should converge to an optimal configuration for future classification, one with the minimum possible error rate. This is analogous to finding the best coefficients for the linear sum of input values to produce the desired output response.

Due to the linear nature of this classification, classes can only be separated by a neuron if they can be physically separated in space by a hyperplane. Image classification data is complex enough that this level of separation is inadequate. For these applications, a non-linear boundary is necessary, and a more complex neural network should be implemented.

6.2 Multi-Layer Neural Networks

While neurons by themselves are not adequate for all classification problems, they can serve as the building blocks for multi-layer neural networks that can handle non-linear classification situations. By chaining together several layers of neurons, a structure called a multi-layer perceptron is formed. These more complex structures are capable of non-linear classifications.

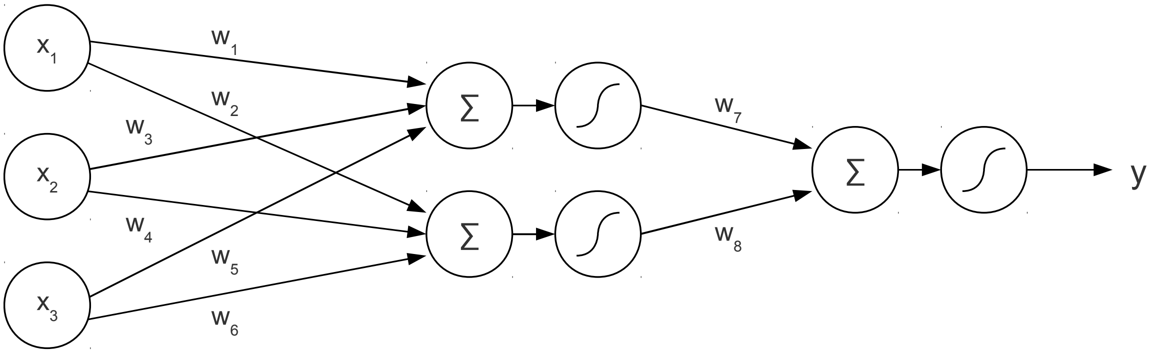


Figure 17: A multi-layer perceptron with three input nodes, two hidden nodes, and one output node. The threshold function used is the logistic function. It is a good approximation of the binary step function, and is differentiable, a key requirement.

A common multi-layer perceptron classifier has three layers – an input layer, a hidden layer, and an output layer. The hidden layer has neurons that each accept weighted sums of the input values and produce output based on a threshold function. The output layer accepts weighted sums of the hidden layer output values and determines the final classifier outputs based on a threshold function. The threshold function for multi-layer perceptrons must be a differentiable approximation of the step function for the purposes of training [16]. For this paper and many others, the logistic function was used.

6.2.1 Multi-Layer Perceptrons for Wafer Map Classification

Two sets of wafer maps were created for developing the classifier, a 50 map training set with 5 maps per class, and a 30 map test set with 3 maps per class. No two maps are identical, and a realistic amount of variation exists within classes. The classes themselves represent a variety of real-world signatures, including scratches, rings, lines, patches, and spokes. As mentioned above, wafer map compression was fixed at 50x50 units, a high enough resolution to preserve signature features without making the neural network too large and cumbersome to train. This resolution turns out to be high enough to approximate chip-level resolution on almost all products, where chip counts are usually less than 2,500.

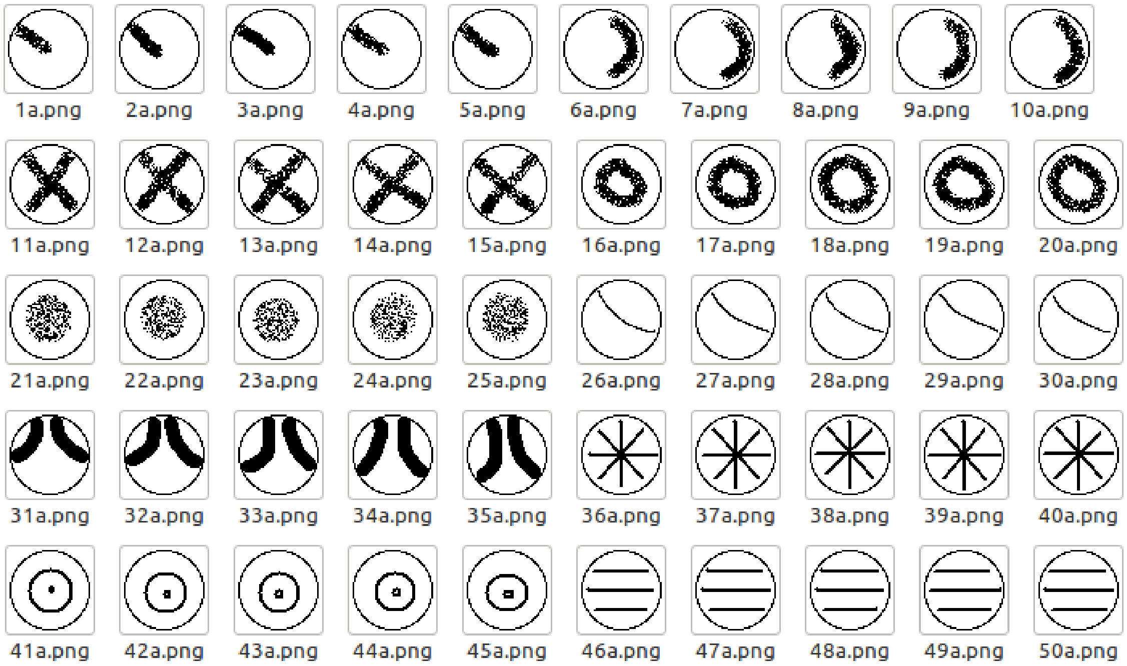


Figure 18: The 50 map training set used for the purposes of this report. There are 10 classes, each with 5 training images. Standard 50x50 map compression was used. No two wafer maps are exactly alike.

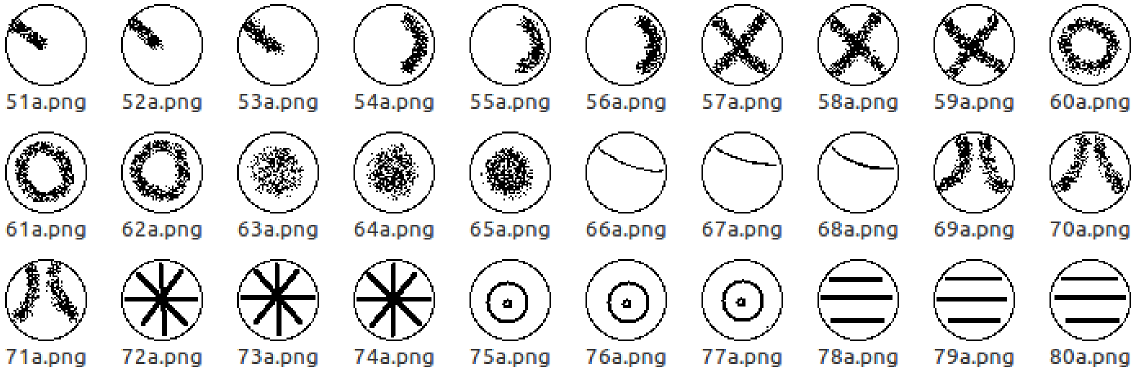


Figure 19: The 30 map test set used to test classification accuracy. Again, no two maps are exactly alike. These maps are also all distinct from the training maps.

6.2.2 Network Input and Output Structure

Determining the proper structure for a multi-layer perceptron is an exercise in model building. Thanks to the power of today's computers, a relatively large neural network was able to be constructed for this report. Instead of computing wafer summary characteristic features or performing feature extraction to reduce the number of inputs, every binary map spatial unit for a signature has its own input to the network. For the standard 50x50 binary compression of a wafer map used in this report, the neural network accepts 2,500 distinct inputs of 1 or 0.

The output layer structure was designed to represent a binary number. This method was chosen instead of creating an output neuron for each unique signature class. By using a binary number, where each digit is represented by an output neuron, large numbers of signatures can be represented. When new classes are added, the structure of neural network remains constant, with only weights being updated. Computation time also decreases because the number of weights that need to be adjusted during back-propagation is lower. For this report, the number of output neurons chosen was 10. This caps the number of simultaneous signatures that can be handled by the classifier at 2^{10} or 1,024. This is far larger than the 10 that could be stored if the output neurons each represented a distinct class. Using a separate output neuron for each class does have the advantage of clearer separation between classes with less training.

6.2.3 Network Hidden Layer Structure

The structure of the neural network's hidden layer is a non-trivial problem. There are two main considerations – the number of layers and the number of nodes. For most applications, no more than two hidden layers are necessary [16]. After experimentation, it was found that a single hidden layer was suitable for wafer map signature recognition. Adding more layers significantly increases model complexity and training time.

The number of nodes in the hidden layer usually falls somewhere in-between the number of input and output nodes [17]. If the number of neurons chosen is too small, the resultant model becomes too vague and may not classify effectively. If the number of neurons is too large, the model may overfit the training set and not handle new data well. A larger number of neurons also naturally increases calculation time to train the network and perform classification, plus results in high initial error and a much longer time to converge to the best set of weights.

For this application, the number of hidden layer neurons was determined by plotting classification accuracy for test data as the number of neurons increased. To determine classification accuracy, the network output was rounded to a binary number using a method described later and then compared against the correct binary value for the class. The accuracy below represents the percentage of the 30 test images that the network classified correctly. As expected, for a given number of training epochs, the accuracy improved as the number of neurons increased, because the model became increasingly representative. However, accuracy leveled off at slightly above 90% with 20 nodes, and no additional benefit was realized as hidden node count increased further.

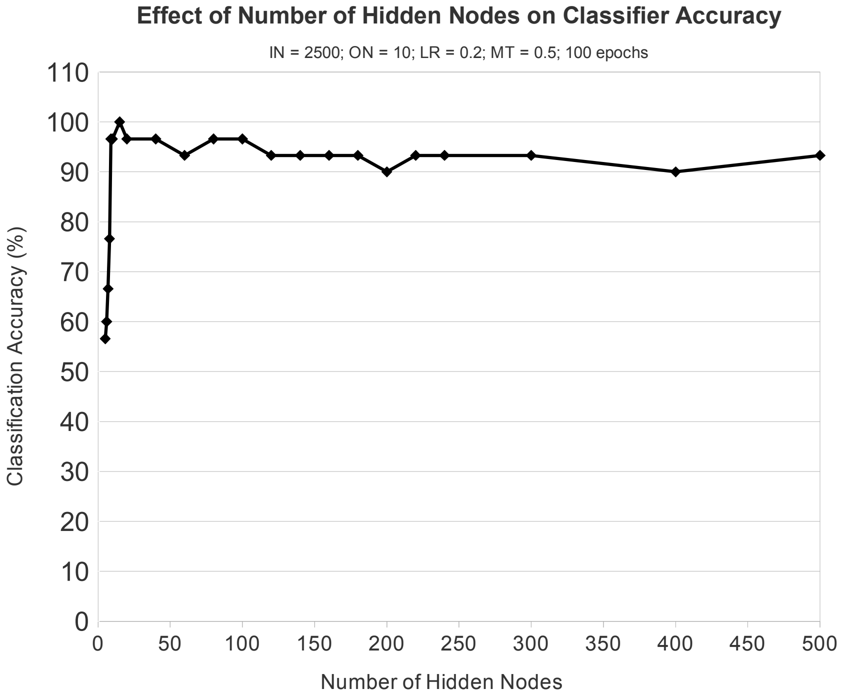


Figure 20: Effect of the number of hidden nodes on classifier accuracy.

In fact, very high node counts should actually result in worse test accuracy. This case, where the model is overfitted to the training set, was not reached during testing. The maximum number of nodes tested was 1000, and came at a heavy computational cost and failure to converge after 100 training epochs.

Hidden node count also affects the training of the neural network. Experimentation showed that as node count increased, the number of epochs necessary to approach to the model's minimum error decreased. However, no real additional improvement was achieved with node counts over 80. At this point the model was sufficiently complex to efficiently represent the training data. After a certain point, initial error, the number of epochs needed to reach a certain error rate, and the computation time per training epoch increased dramatically. Overall time to convergence, including calculation time and number of epochs, increased so much that obtaining results proved infeasible for the purposes of this report.

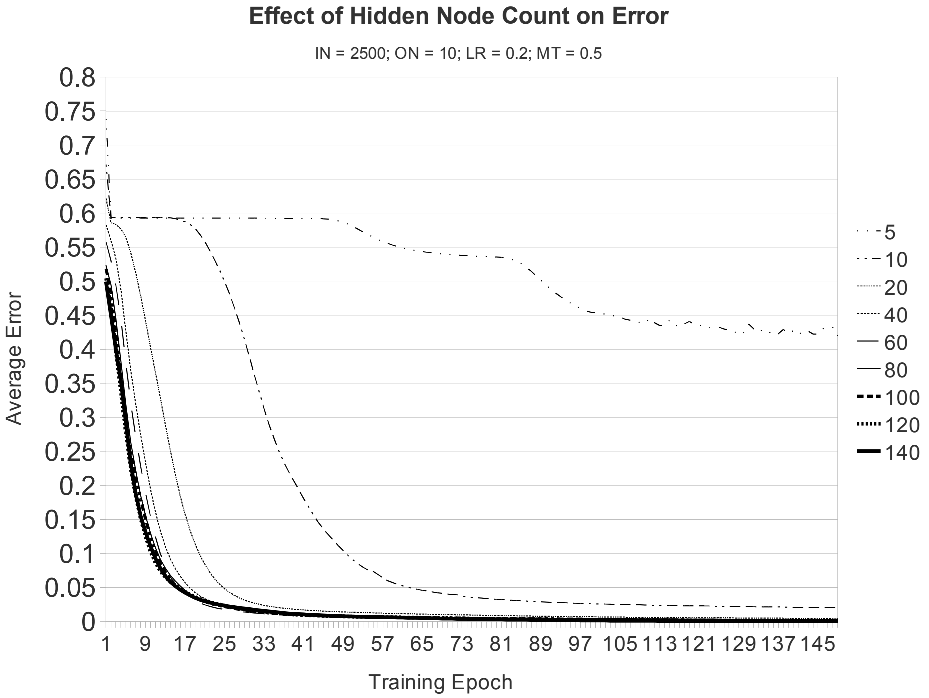


Figure 21: Effect of hidden node count on error.

Time of training and classification was an important consideration in choosing the number of hidden layer neurons. As shown below, for a given number of epochs, the training time increased linearly with the number of hidden layer nodes. For this testing, the number of epochs was fixed at 150. However, as described above, at very high node counts the number of epochs necessary to converge increases significantly. This will affect the rate of increase below.

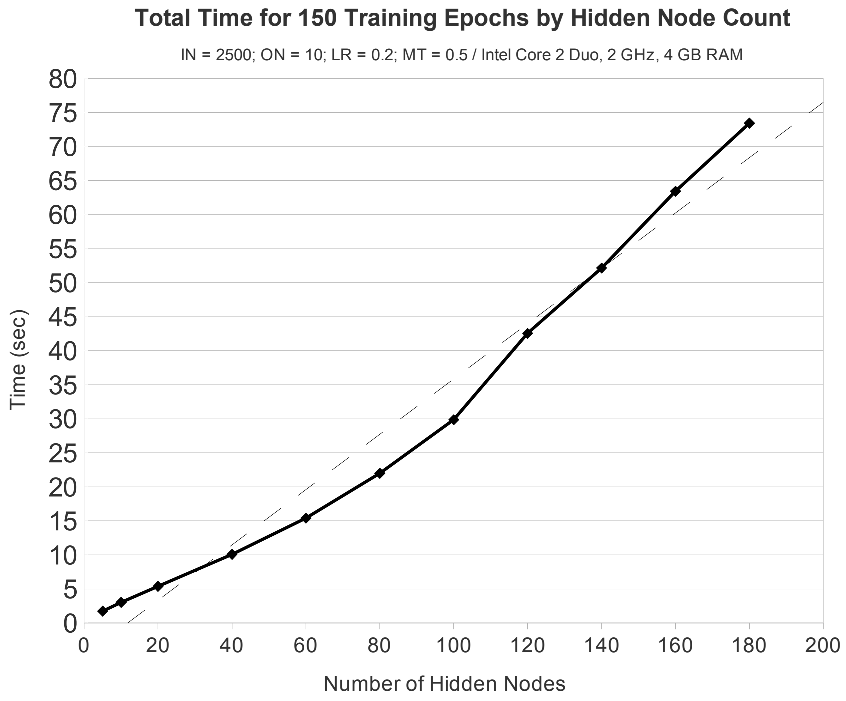


Figure 22: Total time for 150 training epochs by hidden node count.

Based on results for accuracy and convergence rate, and a reasonable 30 second training time for 150 epochs, 100 hidden nodes were used for the majority of this paper's wafer map analysis testing.

6.2.4 Network Training Details

Training of a multi-layer perceptron is accomplished through a process known as back-propagation. Back-propagation uses the technique of gradient descent to find the set of weights that produce the model's global minimum error. As with the single neuron case, training is an iterative process. Each epoch moves the weights closer to their optimal configuration. However, if the model and learning parameters are not set up optimally, the back-propagation algorithm can have difficulty finding the global minimum, or even completely fail to converge.

Like the single neuron case, the weights in a multilayer perceptron are updated according to a learning rate multiplier. This factor affects the speed of training for the network. If the learning rate is very low, the classifier requires more iterations to find the optimal weights for minimal error. If the learning rate is too high, the classifier may fail to find the global minimum of the error, leading to sub-optimal weights and poor classifications. During experimentation, the latter effect was clearly visible. A network with a learning rate of 0.7 did not converge even after several hundred training epochs. Networks with learning rates from 0.1 to 0.4, however, converged after only around 20 epochs. Though not shown, learning rates much smaller than 0.1 did not significantly improve the final classifier error but did result in an increase in the number of epochs required to attain it.

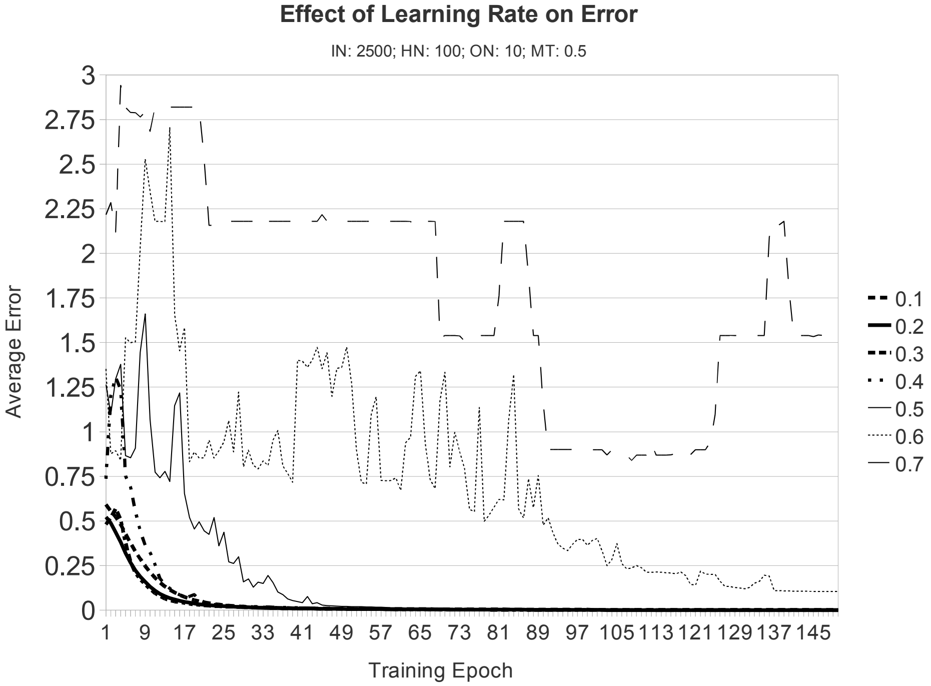


Figure 23: Effect of learning rate on error.

Another factor in neural network training is momentum. Under certain circumstances, the training algorithm can fail to find the correct weights to reach the model's global minimum error. Instead, it can get trapped in a local minimum with sub-optimal weights, affecting classification accuracy. Adding a momentum term to the weight update portion of the back-propagation algorithm helps to avoid this problem. The momentum term adds a fraction of the previous iteration's weight update to the current weight update.

$$w_{new} = w_{old} + delta_{new} + momentum \cdot delta_{old}$$

This addition prevents rapid changes in weight direction, and can serve to carry the algorithm out of a local minimum [18]. Some variations of the algorithm include not only the previous weight update term, but also the term before that, further preventing rapid changes. If the algorithm gathers enough momentum in the direction of the global minimum, the convergence rate can also show noticeable improvement. During experimentation for this report, using a high momentum multiplier of 0.9 resulted in a network that started with a higher initial error but found an acceptable error of 0.1 in approximately six fewer epochs than the case where the momentum multiplier was only 0.1.

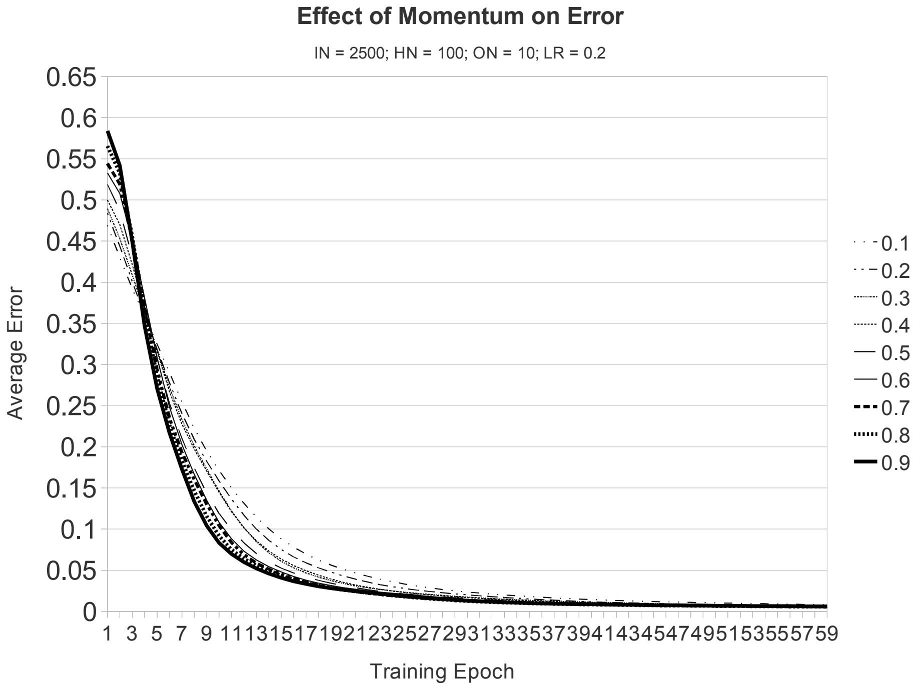


Figure 24: Effect of momentum on error.

The effects of momentum on the map-to-map classifier weight updates can be seen clearly in the figure below. The upper-left case with no momentum term showed lower initial error but high variation in error as each new map was introduced. The lower-left case with a 1.0 momentum multiplier showed significantly less variation in error from map-to-map, since the change in weights at each iteration were an equal combination of the current update and previous iteration's update.

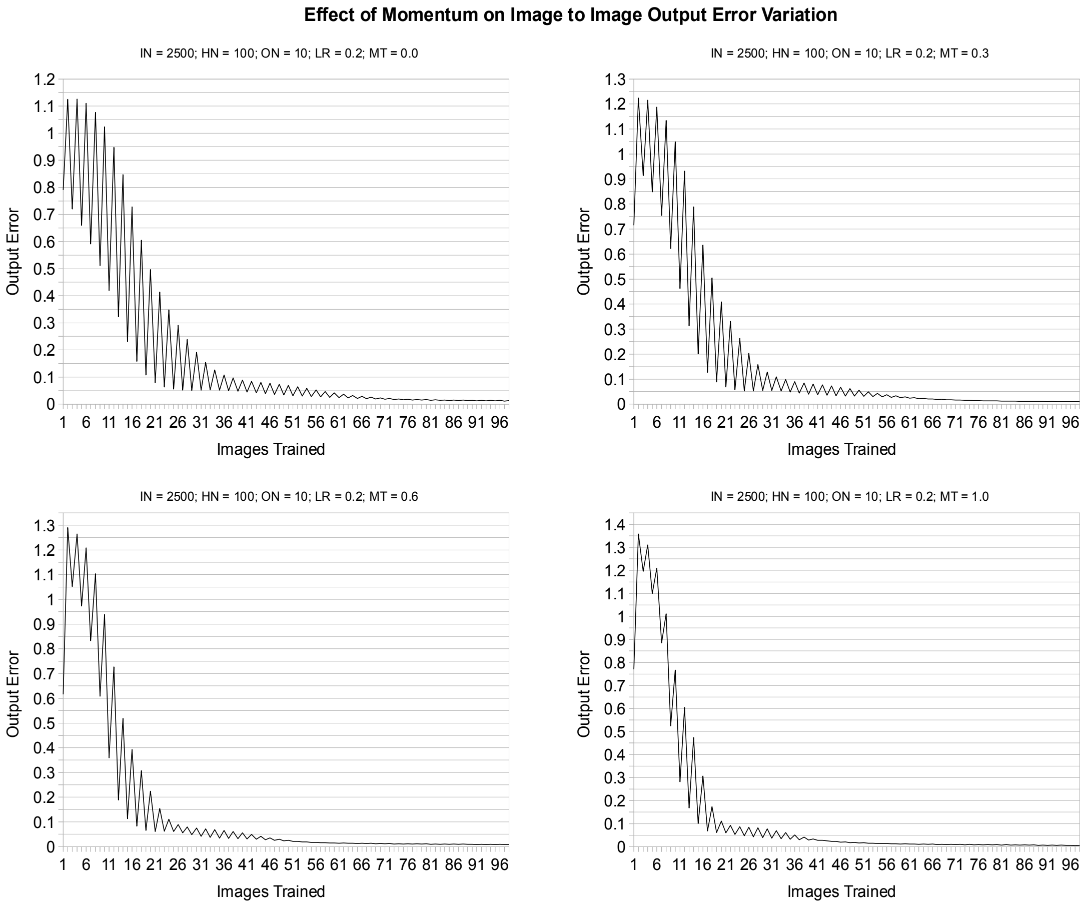


Figure 25: Effect of momentum on image to image output error variation.

In an application for wafer maps, where new maps are introduced one-by-one, less variation in weights from one image to the next is desirable. In testing with the map data set shown above, a moderate momentum multiplier of 0.5 was chosen to reduce the magnitude of the weight swings and help the algorithm avoid local minima that may develop as new maps are added to the training set.

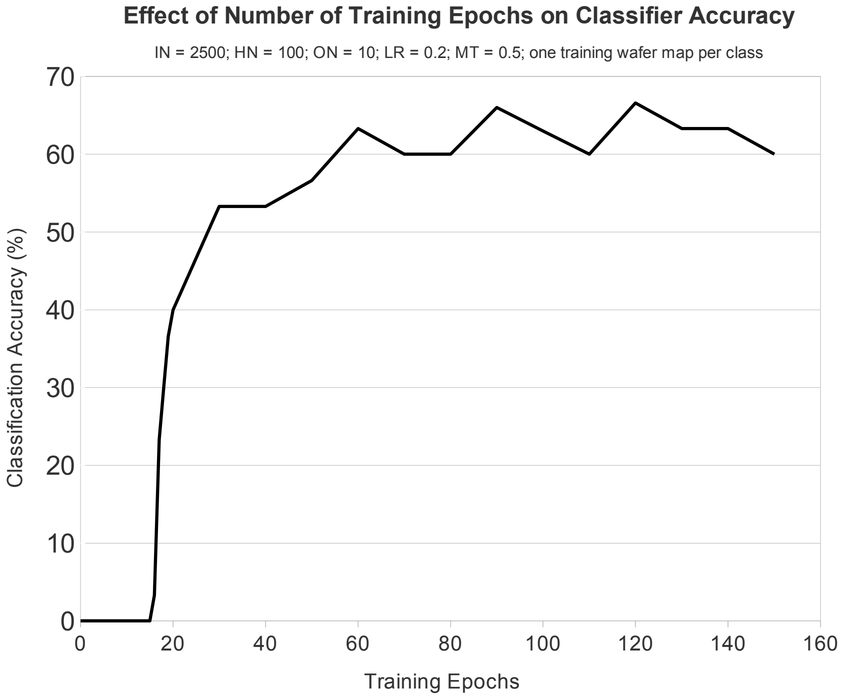


Figure 26: Effect of number of training epochs on classifier accuracy.

Another question for neural network training is how many epochs are necessary to achieve good classification accuracy. For this report, neural network structure was fixed at 2,500 input nodes, 100 hidden nodes, and 10 output nodes as described above, and back-propagation parameters such as learning rate and momentum were set at 0.2 and 0.5, respectively. The number of training epochs used to teach the network ten classes, each with one training map, was varied and the average classification error on the 30 training images was measured. The results showed that for this neural network configuration, approximately 60 epochs, or six per class, were sufficient to attain the maximum accuracy possible.

As can be seen in the figure above, classification accuracy during experimentation peaked at just over 65%. This is because the classifier had only one training map per class. The figure below shows that increasing the number of unique maps trained for each class reduces the final test data classification error.

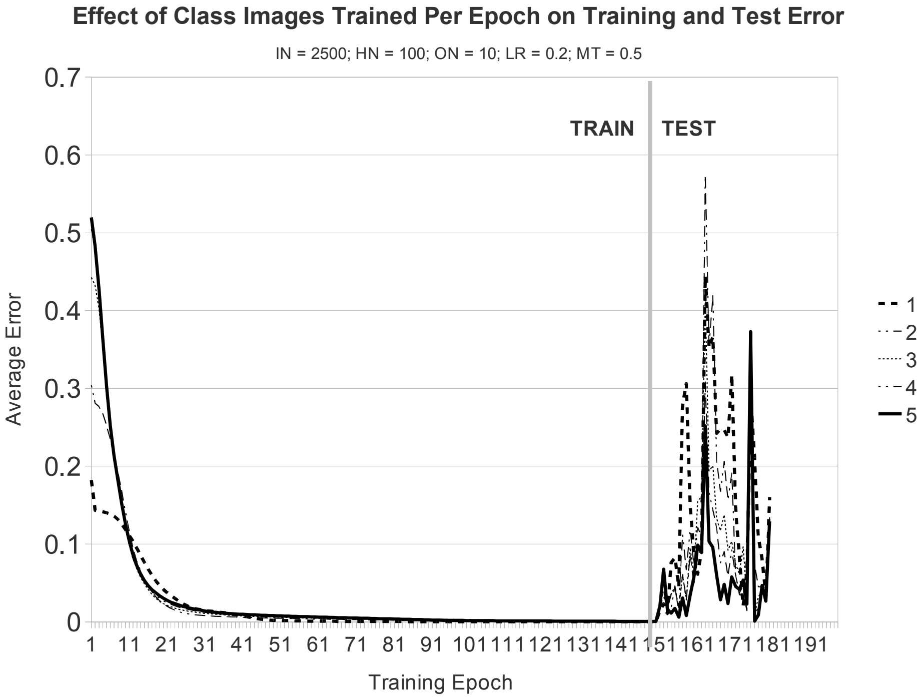


Figure 27: Effect of class images trained per epoch on training and test error.

The test data accuracy is shown in another way in the figure below. As seen earlier, training with only one map per class results in approximately 60% accuracy. However, increasing the training set to five maps per class raises accuracy to over 90% for the same number of epochs. Clearly, as more confirmed wafer map data is added to the classifier, detection ability increases substantially. This is because the network is better prepared to handle variation in the signature from map to map. Training the classifier with rotated versions of the signature, described earlier, is one component of this strategy. Instead of leaving the network unprepared for rotation, it is taught in advance, significantly improving accuracy.

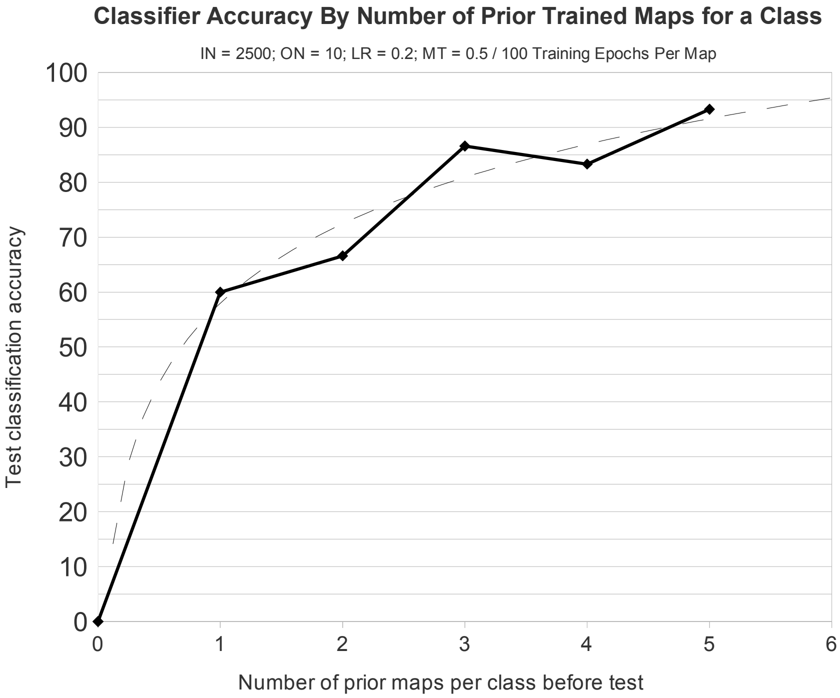


Figure 28: Classifier accuracy by number of prior trained maps for a class.

While increasing the number training epochs and unique maps trained per class both improve accuracy, increasing the number of distinct classes does not. As can be seen below, the number of epochs necessary to reach a low error rate increases with each additional class added. While the number of training epochs chosen for testing earlier, 60, is overkill for one or two classes where convergence occurs in under 10 epochs, it is suitable for larger quantities of classes. Testing with a maximum set of 1,024 classes should be done to determine the final minimum number of epochs necessary, but that quantity of signature data was not available for this report.

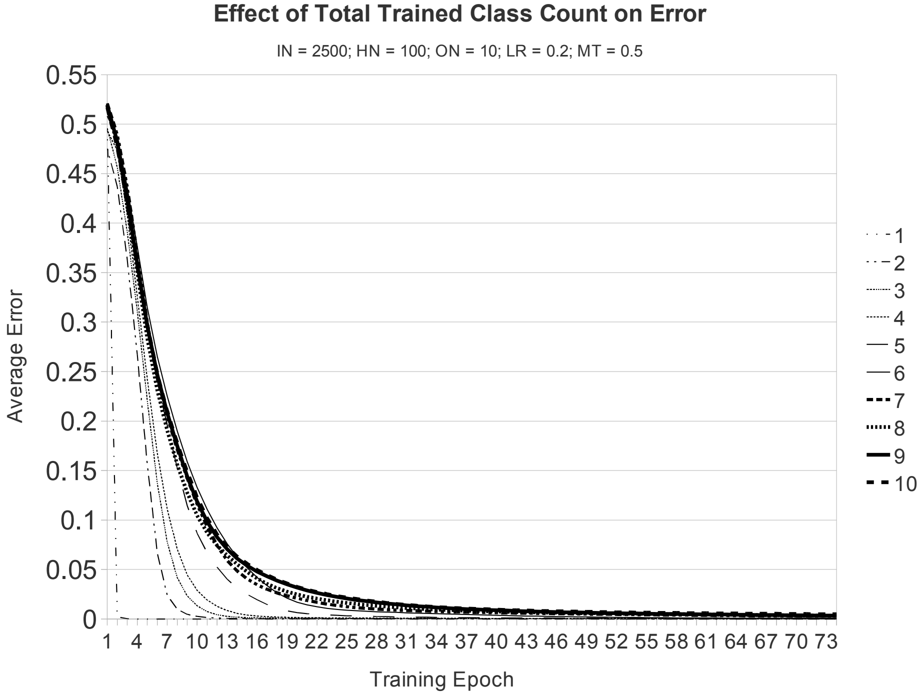


Figure 29: Effect of total trained class count on error.

6.2.5 Initial Weight Configuration

During testing a strong dependence on the initial weight selection was noticed. At first, weights were chosen randomly in the interval (0, 1]. However, this produced networks which rarely converged, no matter how learning rates, momentum multipliers, and hidden node quantities were configured. Weights oscillated wildly, and rare cases of convergence occurred only after thousands of epochs. Simply changing the starting random interval to [-0.5, 0.5] immediately corrected this problem. The new networks were able to attain low error rates even after only a handful of epochs, as shown in the many examples above.

6.3 Classification Determination

After passing through the neural network, each signature's output needs to be compared with known classes. Each output neuron of a perfect classification should result in exactly one or zero. Instead of using a more computationally intensive method such as k-nearest neighbors between the sample point and known class points, the knowledge that each output must either be a one or zero is used. The sample bits are either rounded up to one or rounded down to zero, depending on how close they are to either number. The amount needed to round the number is stored for each bit. From this rounding a clear binary code is created, and the accompanying sum of squared errors can be used to determine if the sample point required too much rounding to match the known class. If it did, then the sample point is assigned the next available binary signature classification number, and the error between the original sample point bits and the new binary value is back-propagated multiple times through the neural network for training. If the sample point was within the error threshold, then it is classified as the signature. In this case also the error from the class is back-propagated through the network to strengthen the classifier.

Chapter 7

Commonality

After passing all maps through the neural network, a set of signature groups is obtained. Each group may contain multiple submaps with similar signatures, or rotated versions of the same signature. The question now is what fab process caused each issue. This is where the commonality tool is useful.

7.1 Overview

The commonality is a relatively simple function. Signature groups from the neural network classifier are taken and their fab processing histories analyzed. If all signature occurrences from a group ran on a particular piece of fab equipment at a certain step, suspicion falls on that tool as the source of the problem. The run portion of the suspicious tool at the step is also important, as discussed earlier. The commonality tool finds the percentage of signature occurrences that ran on each step-equipment combination in the fab.

The results are first sorted by this percentage, from high to low. Thus, 100% commonalities, where all signatures from a group ran on the same step-equipment pair, are at the top of the list. The list has secondary sorting on the run percentage of the tool at the step, from low to high. If, for example, five step-equipment pairs show 100% commonality, the one which has a tool that only runs 20% of the material at its step will be shown first, followed by one that runs 40%, and so forth. With this sorting, the strongest commonalities are shown first.

STEP	TOOL	AFFECTED	TOTAL	COMMON	PORTION	TIMEGAP
MEM055909	CVD051	6	6	100.00%	26.00%	16
MEM072100	ETCH05	6	6	100.00%	78.00%	30
MEM140200	ETCH13	5	6	83.33%	21.00%	23
MEM162510	PHTO12	4	6	66.67%	36.00%	41
MEM065120	METL06	4	6	66.67%	89.00%	10
MEM040100	CVD011	3	6	50.00%	66.00%	14

Figure 30: A sample commonality output. The tool CVD051 running at step MEM055909 is the most suspicious, since it ran 100% of the wafers with a particular signature yet only 26% of material overall at the step. It also ran the affected material within a relatively short 16 hour period, increasing the likelihood that it is the source.

A final piece of information in the commonality is the time gap measure. The time gap for a step-equipment pair is the number of hours between when the first affected wafer ran and when the last affected wafer ran. Smaller time gaps indicate that the affected material all ran within a short time period, increasing the level of suspicion for the pair. This is the final sorting column, but since run portions are rarely equal, it does not usually affect the rankings.

7.2 Implementation

The commonality function used during testing for this report was developed exclusively in SQL. SQL provides powerful aggregation functions that are ideal for analyzing groups. The database is set up with two tables. The first, called MAP_SET, has a list of all maps and the signature classes they belong to. Note that the identification number used, WAFER_ID, is the original wafer identification number. Therefore, if a wafer has multiple signatures, it can appear several times in this table, once for each separated signature. The second table, called RUN_HISTORY, has a list of all wafers and the tool they ran on at each process step. Using this configuration, finding base commonalities is as simple as using the following SQL query.

```
SELECT      R.STEP, R.EQP, COUNT(M.WAFER_ID) AS MAP_COUNT
FROM        MAP_SET M, RUN_HISTORY R
WHERE       R.WAFER_ID = M.WAFER_ID
            AND M.SIGNATURE_ID = 0100101000
GROUP BY    R.STEP, R.EQP
ORDER BY    MAP_COUNT DESC
```

Figure 31: SQL code to generate the commonality count for each step/equipment pair.

This statement pulls the complete run history for all maps in the 0100101000 signature group. It then summarizes the data using the aggregate function **`COUNT`**. The result is a list of the number of affected maps that ran on each step/equipment combination. Step/equipment pairs that have the same map count as the total number of maps in signature 0100101000 are 100% commonalities.

From here, additional data can be joined to this base table that increases the usefulness of the commonality. These tables can also be easy generated in SQL. For example, run portion is accomplished with the following statement.

```
SELECT      T1.STEP, T1.EQP, T1.UNIT_COUNT/T2.TOTAL_COUNT AS PORTION
FROM        (SELECT     STEP, EQP, COUNT(WAFER_ID) UNIT_COUNT
            FROM        RUN_HISTORY) T1
JOIN        (SELECT     STEP, COUNT(WAFER_ID) TOTAL_COUNT
            FROM        RUN_HISTORY) T2
ON          T2.STEP = T1.STEP
```

Figure 32: SQL code to generate a run portion table for each step/equipment pair.

Two subtables are created, one with the count of wafers for each step/equipment pair, the other with the count of total wafers run at each step. The first count is divided by the second count to get PORTION after the tables are joined on STEP.

Time gap can be computed similarly. Once all three tables are created, they can be joined on STEP and EQP to obtain the table shown in Figure 30. The results can be written to a third database table, keyed by the signature ID, and retrieved later when the results of the wafer map analysis tool are shown to the reviewing engineer.

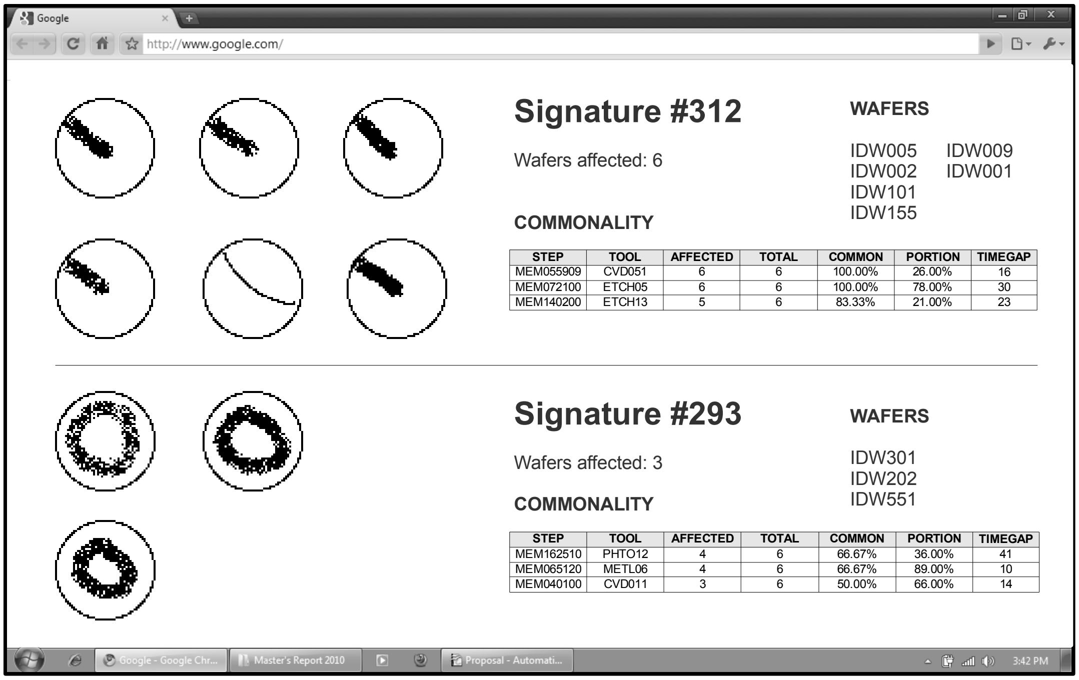
Chapter 8

Next Steps

8.1 Integration

This report has covered the various components of a possible wafer map analysis system, including data preparation, signature separation, classification, and commonalities. Each was tested and optimized independently using representative data, but work continues to integrate these functions into a single application that can be scheduled to run daily.

After the wafer map analysis is complete, the application needs a way to present results to the reviewing engineer. For convenience, a web interface is under development. This interface shows signature maps grouped by class along with commonality results. From this report the engineer can quickly review all signatures found by the algorithm and see initial commonality results. Errors in the classification and commonality will not prevent the tool from being useful. If nothing else, the tool can highlight signals that otherwise would have been missed, providing an invaluable starting point for investigation.



Signature #312

Wafers affected: 6

WAFERS

IDW005 IDW009
IDW002 IDW001
IDW101
IDW155

COMMONALITY

STEP	TOOL	AFFECTED	TOTAL	COMMON	PORTION	TIMEGAP
MEM055909	CVD051	6	6	100.00%	26.00%	16
MEM072100	ETCH05	6	6	100.00%	78.00%	30
MEM140200	ETCH13	5	6	83.33%	21.00%	23

Signature #293

Wafers affected: 3

WAFERS

IDW301
IDW202
IDW551

COMMONALITY

STEP	TOOL	AFFECTED	TOTAL	COMMON	PORTION	TIMEGAP
MEM162510	PHTO12	4	6	66.67%	36.00%	41
MEM065120	METL06	4	6	66.67%	89.00%	10
MEM040100	CVD011	3	6	50.00%	66.00%	14

Figure 33: An illustration of the possible final output for the under-development web-based results interface.

8.2 Handling Other Types of Maps

A close parallel to defect maps is the final electrical test map generated after all wafer processing is complete. As each chip undergoes various electrical tests, it is placed into one of a set of bins based on its performance. For example, if a critical voltage parameter on the chip is out of the specification window, the chip may be classified as a certain bin. If the chip fails a particular core operation, such as reading data from its memory, it may be classified as another bin. Once a chip fails for a particular bin, it cannot fail for another.

Electrical bin maps end up being a rich composite of failures from all steps in the fab. Their major advantage is that every wafer is tested and produces a map, while defect scans sample only a small percentage of material. Their main disadvantage is the time from point of failure to detection – if a problem happens early during wafer processing, it can be weeks before affected material finishes production and the failure shows up at electrical testing.

The techniques discussed in this paper for defect maps could easily be applied to electrical bin maps. Instead of creating submaps by defect size, they would be created by separating different bins. Defect density overlap modifications to the agglomerative clustering algorithm would not be necessary, since different overlapping signatures would likely fail for different bins and already be separated. Bin submaps would already be in binary format – pass or fail. The only problem would be that it would be impossible to create a general coordinate system across products, because the bin failures are already on a product-specific chip level. However, with data available for every single wafer, enough information would likely be available within each product to determine the root source of the problem.

8.3 Self-Training

Given a probable signature group and similar maps that didn't quite pass the threshold for identification, the results of a commonality can be used to either show a connection between the questionable wafers and the main group or a clear disconnect. If borderline maps ran on the affected tool during the suspect time period, the algorithm can assume that they were from the same issue and automatically feed them back for training as the main class. This would allow the classifier to improve its robustness to signature variation without any human feedback. If the borderline maps did not run on the affected tool during the suspect time period, the classifier can have more confidence that those maps in fact belong to a separate issue. This sort of intelligent auto-training could be implemented in a future version of the wafer map analysis system.

8.4 Core Algorithm Improvements

Finally, many opportunities exist to improve the core algorithms used above. For example, the quartile-based noise cluster removal strategy discussed in Section 5.1.2 is primitive compared to methods such as the Hierarchal Density Shaving algorithm presented in [19]. Also, in practice, neural networks are rarely given a full map as an input. This approach creates larger-than-necessary networks with poor generality. Better approaches exist to extract important features from the data first and then use only these as the neural network inputs. Lastly, the classification determination algorithm would benefit from a switch to probabilistic methods such as those used in [7].

Chapter 9

Conclusion

Semiconductor fabs are some of the most expensive manufacturing facilities in the world. Maximizing wafer yield is one of the best ways to recoup the investment made in the facility and its equipment. Fabs produce massive amounts of data that can help improve yield, including defect scan maps, yet engineers and technicians have time to review only a small fraction of it. Therefore, software tools that help search for signals in the data are invaluable in the industry.

This report investigated the use of several popular algorithms for wafer defect map signature classification. These algorithms were adapted to fit the unique characteristics of defect data, and the results overall were promising. In particular, the modified agglomerative clustering method showed good ability to separate signatures, even overlapping ones, and the large neural network was able to attain over 90% accuracy with only a handful of training maps. The addition of an automatic commonality component only serves to increase the immediate usefulness of the grouping results. Significant work remains to be done, but a starting point for automatic defect map analysis has been established.

Bibliography

[1] "WSTS Semiconductor Market Forecast Spring 2010." *World Semiconductor Trade Statistics*, 8 June 2010 <http://www.wsts.org>.

[2] “GLOBALFOUNDRIES Takes Final Steps Toward Fab 2 Groundbreaking.” *Globalfoundries, Inc.* 9 June 2009 <http://www.globalfoundries.com/newsroom/2009/20090609.aspx>

[3] Gruber, Harald. “The Semiconductor Industry Review of the Sector and Financing Opportunities.” *European Investment Bank*, 28 October 2008

[4] Dannen, Chris. “Intel v. ARM: The Battle to Run Your Smartphone and Netbook.” *Fast Company*, 23 March 2009 <http://www.fastcompany.com/>

[5] “Defect Signature Analyzer (DSA) Datasheet.” *SiGlaz, Inc.* <http://www.siglaz.com/products/datasheet/dsa.pdf>

[6] Chang, Kyung-Soo, Chi-Hyuck Jun, and Sang-Ho Lee. “A Rule-Based Recognition of Spatial Defect Patterns on Semiconductor Wafers.” *Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference,* 17-20 Dec 2006, p.1304-1310.

[7] Karnowski, Thomas, Shaun Gleason, and Kenneth Tobin. “Automatic Classification of Spatial Signatures on Semiconductor Wafermaps.” *SPIE 22nd Annual International Symposium on Microlithography,* 9-14 March 1997.

[8] Borisov, Alexander, Eric St. Pierre, and Eugene Tuv. “Spatial Patterns in Sort Wafer Maps and Identifying Fab Tool Commonalities.” *2008 IEEE Advanced Semiconductor Manufacturing Conference*, 2008, p.268-272

[9] Chen, Fei-Long, and Shu-Fan Liu. “A Neural-Network Approach to Recognize Defect Spatial Pattern in Semiconductor Fabrication.” *IEEE Transactions on Semiconductor Manufacturing*, Vol.13, No.3, August 2000, p.366-373

[10] Huang, Cheng-Lung and Te-Sheng Li. “Defect Spatial Recognition Using a Hybrid SOV-SVM Approach in Semiconductor Manufacturing.” *Expert Systems with Applications*, 2009, p.374-385

[11] Walfish, Steven. “A Review of Statistical Outlier Methods.” *Pharmaceutical Technology*, 6 Nov 2006

[12] Hastie, Trevor, Robert Tibshirani, and Guenther Walther. “Estimating the Number of Clusters in a Data Set via the Gap Statistic.” *Journal of the Royal Statistical Society*, Vol. 63, Part 2, 2001, p.411-423

[13] Cardie, Claire, Seth Rogers, Stefan Schroedl, and Kiri Wagstaff. "Constrained K-means Clustering with Background Knowledge." *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, p.577-584.

[14] Manning, Christopher, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Informational Retrieval*, Cambridge University Press, 2008, p.384-385.

[15] "Rotation Matrix." *Wolfram MathWorld*. <http://www.wolfram.com>

[16] Beale, Russell and Tom Jackson. *Neural Computing: An Introduction*, CRC Press, 1990, p.73.

[17] Keller, Paul and Kevin Priddy. *Artificial Neural Networks – An Introduction,* SPIE Publications, 2005

[18] Zaknich, Anthony. *Neural Networks for Intelligent Signal Processing*, World Scientific, 2003, p.147.

[19] Ghosh, Joydeep, Gunjan Gupta, and Alexander Liu. "Automated Hierarchical Density Shaving: A Robust Automated Clustering and Visualization Framework for Large Biological Data Sets ." *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* Vol. 7, No. 2, April-June 2010, p.223-237.