Unsupervised Wafermap Patterns Clustering via Variational Autoencoders

Peter Tulala\textsuperscript{1}, Hamidreza Mahyar\textsuperscript{2}, Elahe Ghalebi\textsuperscript{3}, Radu Grosu\textsuperscript{4}

\textit{Institute of Computer Engineering, Vienna University of Technology (TU Wien)}
\texttt{\{ptulala, hmahyar, eghalebi\}@cps.tuwien.ac.at, radu.grosu@tuwien.ac.at}

Abstract—Semiconductor manufacturing processes are prone to process deviations or other production issues. Quality assurance of every processing step and measuring wafer test values is crucial for finding possible root causes of these problems. Automated visual inspection and recognition of patterns in wafermap data obtained during different processing steps has a potential to significantly improve the efficiency of finding early production issues and even help with adjustment of the production parameters to automatically resolve them. In this paper, we present a machine learning approach for unsupervised clustering of spatial patterns in wafermap measurement data. Measured test values are first pre-processed using some computer vision techniques, followed by a feature extraction based on variational autoencoders to decompose high-dimensional wafermaps into a low-dimensional latent representation. Final step is to detect the structure of this latent space and assign individual wafers into clusters. We experimentally evaluate the performance of the proposed method over a real dataset.

Index Terms—Wafer test data, Image processing, Semiconductor manufacturing, Pattern recognition, Variational Autoencoder, Clustering

I. INTRODUCTION

New challenges outlined by the “Industry 4.0” [14], such as the requirement for producing smaller lot sizes with highly customer-specific products, are propelling existing industries to adapt their manufacturing processes to new paradigms in order to sustain competitiveness in increasingly global markets. Automated root cause analysis and decision making with reduced human intervention has potential to significantly improve manufacturing efficiency of semiconductor manufacturing. Algorithms for detecting defects from given sensory data are valuable contribution in accordance with this goal.

Manufacturing defects exhibit typical shapes in measured wafer test data, e.g. rings, spots, repetitive patterns or scratches. Recognizing these patterns is an essential step for finding root cause of production issues. Several methods based on traditional image processing approach have been proposed [4] [10].

More robust methods utilize some machine learning techniques to recognize more complex patterns in wafer test data. There have been proposed methods based on supervised training of mixture models [17], neural networks [7] and support vector machines [6]. Although these methods are powerful, their supervised nature still requires a human expert to craft a training dataset with manually labelled data.

The apparent advantage of unsupervised pattern detection approach lies in an elimination of subjective factors from pattern recognition task, which in turn reduces costs and number of classification errors. The hidden dependencies between different types of wafer defects are detected automatically without intervention of human expert which enables detection of patterns that were unknown or overlooked before. This type of approach includes self-organizing neural networks [5], self-organizing maps [19] as well as techniques based on dimensionality reduction like diffusion maps [18] and discriminant analysis [25].

In this paper, we propose an unsupervised method for clustering of wafermap patterns based on variational autoencoder. The algorithm consists of three consecutive steps. In the first step, the raw wafer dataset is pre-processed using some computer vision techniques. Variational autoencoder [23] is then trained on cleansed wafermaps to extract latent features. The last step applies $k$-means clustering on latent features to group wafers based on a distant measure. In the Section V, we compare some commonly used feature extraction methods and we show that the method based on variational autoencoders yields better separated clusters with more structure in majority of cases when different number of clusters or size of latent space is considered.

II. DATA PRE-PROCESSING

Data pre-processing is a crucial step addressing several data quality issues before applying the machine learning algorithm. The main goal of this step is to ensure that individual wafer measurements are comparable – especially by finding and removing outliers, imputing missing measurements and data normalization [8]. The result of data pre-processing step is a cleansed dataset that can be used for further feature extraction and classification tasks. The overall procedure is depicted in Figure 1. The result of the preprocessing is depicted on a sample wafer in Figure 4.

A. Wafer test data

Wafer data are stored in CSV files, where each line of the file represents a single chip. Position of the chip within a wafer is stored as a tuple with coordinates and individual test values are stored in corresponding columns as floating point numbers. We have found that treating each test measurement as a bitmap

\* Authors contributed equally
is more suitable for the purpose of finding spatial patterns in the wafer test data.

Our available dataset consists of 6 wafer lots (CSV files) with 21 different tests (dimensions). Together we have available 5,964 wafermaps, each one represented as a bitmap of size 193x115px.

**B. Wafer clipping mask**

Wafers maps have irregular shape with some missing values (holes) within the wafer area caused during the measurement of test data. This step creates a clipping mask of the wafer without holes, so that we can address the missing values within the masked area in the data imputation step.

Firstly, we binarize the wafer by replacing all present values with 1 and all missing values with 0. Mathematical morphology approach is then used to close small holes within the wafer. All binary morphological operations are the result of combining two morphological operators – erosion (denoted by $A \ominus B$) and dilation (denoted by $A \oplus B$) where $A$ is a binary image in an Euclidean space and $B$ is a structuring element [22]. The former operation (also called Minkowski addition) is sufficient for closing the holes in the binary clipping mask. Two additional operations from set theory are used in the algorithm – $A \cap B$ (intersection) and $A^C$ (complement). Given two sets of vectors in Euclidean space $A$, $B$, we define:

$$A \oplus B = B \ominus A = \{a + b \mid a \in A, b \in B\}$$  \hspace{1cm} (1)

As shown in Algorithm 1, repeated dilation of the outer area of the wafer is then used to find contours of the wafer.

**C. Removing outliers**

Real-world data are contaminated with measurement errors. Although random and systematic measurement errors caused by physical limitations of manufacturing device lower the quality of recorded data, they are usually within certain range and do not cause major problems when building a predictive model. However, occasional large inaccuracies or malfunctions in measurement process can introduce gross errors called outliers or anomalies.

Outliers may be informally defined as observations that are too distant from our expectations. Many predictive models are based on estimating probability distribution of the data samples. Yet even a single outlier can completely change the characteristics of the the probability distribution. Hence, removing these outliers is often an essential step for many statistical predictive models.

In our approach, we utilize a modified $Z$-score to detect outliers. Given a values of a wafer measurements $X = (x_1, \ldots, x_n)$, the formula for $Z$-score for $x_i$ is defined as:

$$z_i = \frac{x_i - \mu(X)}{\sigma(X)}$$  \hspace{1cm} (2)

The distribution of $Z$-score has mean 0 and standard deviation 1, which means it removes the effects of scale and location from the original dataset. The idea is to identify all measurements with $|z_i| > \lambda$ as outliers for some cutoff threshold $\lambda$.

However, identifying outliers with the Z-score is problematic, because the mean and the standard deviation themselves are highly affected by outliers. To make this method robust, Iglewicz and Hoaglin [13] developed a median-based outlier detection by modifying the Z-score method. The location parameter is replaced by median and the scale parameter is estimated using median absolute deviation (MAD) as:

$$\hat{\sigma}(X) = k \cdot MAD$$  \hspace{1cm} (3)

where $k$ is a constant scale factor and the median of the absolute deviations (MAD) is defined as:

$$MAD = \text{med}(|x_i - \text{med}(X)| \mid x_i \in X)$$  \hspace{1cm} (4)

For normal distribution, the scale factor is defined as:

$$k = 1/\phi^{-1}(3/4) \approx 1/0.6745 \approx 1.4826$$  \hspace{1cm} (5)

where $\phi^{-1}$ is the inverse of cumulative distribution function of the normal distribution (also called probit function), i.e. $\phi^{-1}(3/4)$ is the 75th percentile of normal distribution.

The modified $Z$-score is then defined as:

$$|z_i'| = \frac{|x_i - \text{med}(X)|}{\text{MAD}} = \frac{\phi^{-1}(3/4) \cdot |x_i - \text{med}(X)|}{\text{MAD}}$$  \hspace{1cm} (6)

where values with $|z_i'| > \lambda$ are considered outliers, with the threshold $\lambda$ typically set to a constant cutoff value $\lambda = 3.5$. 

---

**Algorithm 1** Region filling algorithm for binary bitmap of size $m \times n$ represented by the set of vectors $A$

1. $T := \{(a, b) \mid a \notin [0, m], b \notin [0, n]\}$  \hspace{1cm} $\triangleright$ Image boundary
2. $S_k := \{(0, 0), (0, \pm 1), (\pm 1, 0)\}$  \hspace{1cm} $\triangleright$ 4-neighborhood structuring element
3. **repeat**
4. $T := (T \ominus S_k) \cap A^C$
5. **until** convergence
6. **return** $T^C$
This method works well for symmetric distributions, however it is better to use MAD (as seen in Figure 2 for skewed distributions) calculated independently for data points greater than (resp. less than) or equal to median as:

\[
\text{MAD}_L = \text{med}\{x_i | x_i \leq \text{med}(X)\} - \text{med}(X) \tag{7}
\]

\[
\text{MAD}_R = \text{med}\{x_i | x_i \geq \text{med}(X)\} - \text{med}(X) \tag{8}
\]

**D. Imputing missing values**

Missing values within the area of the clipping mask created in Section II-B are replaced with substitute values reconstructed from information present in neighborhood of each missing region. We have utilized Chui-Mhaskar inpainting algorithm based on solving biharmonic equations [9]. Detailed description of this algorithm is out of scope of this paper, we have used an existing implementation provided by scikit-learn library [20] in our experiments.

The wafer image is then scaled to interval \([0, 5] \subseteq [0, 1]\) as:

\[
f(x) = c_1 \cdot (x - \min) + c_2 \tag{9}
\]

where \(c_1 = 0.95\) and \(c_2 = 0.05\).

Finally, all values outside of the clipping mask (i.e. wafer background) are replaced with constant 0.

**E. Denoising**

The last step of the wafer pre-processing is a smoothing of the wafer by reducing stochastic noise. We have used a simple median filtering procedure. Each pixel of the input image \(f\) is iteratively processed by a sliding window \(W\) of size \((k, l)\), producing smoothened image \(g\). This overall procedure can be formulated [15]:

\[
g(x, y) = \text{med}\{f(x - k, y - l) | (k, l) \in W\} \tag{10}
\]

where \(g(x, y)\) represents a point at position \((x, y)\) in output image \(g\) and \(f(x - k, y - l)\) is a point in the input image \(f\).

We have used sliding window of size \(3 \times 3\), that is \(k = l = 3\) in (10). One sample iteration of median filtering is depicted in Figure 3.

![Fig. 3: One iteration of median filtering algorithm. Current data point with value 21 is replaced with value 5, the median value within the \(3 \times 3\) sliding window.](image)

**III. Feature extraction**

The approach used for this task is based on auto-encoding variational Bayes [16]. The dataset obtained by pre-processing can be seen as \(N\) individual dataset consisting of independent identically distributed samples \(x_1, \ldots, x_N\) of a random discrete variable \(X\). We assume that the dataset was generated by a random process that involves an unobserved random variable \(Z\) (called a latent variable). The goal of the feature extraction described in this section is to overcome the curse of dimensionality [2] for the classification task by efficiently approximating posterior inference of the latent variable \(Z\) given an observed value of \(X\).
A. Encoding variational Bayes

We assume $Z$ has a prior distribution $p(z)$ (in our case unit Gaussian $p(z) = \mathcal{N}(0, I)$) and $X$ is conditioned on latent variable $Z$ with likelihood $p_b(x|z; \theta)$, which gives us the latent variable model:

$$p_b(x, z) = p_b(x|z)p(z) \quad (11)$$

Bayesian inference of the latent features directly from posterior $p_b(z|x)$ involves computing marginal likelihood (evidence):

$$p_b(x) = \int z p_b(x|z)p(z)dz \quad (12)$$

which is intractable for complicated distributions.

Instead, a new conditional distribution $q_b(z|x)$ is introduced as an approximation of the intractable $p_b(x|z)$. We assume the posterior $q_b(z|x)$ is a multivariate Gaussian with a diagonal covariance $\mathcal{N}(\mu, \sigma^2 I)$ and is referred to as encoder or recognition model, since it encodes a given true sample $x$ into a probability distribution of possible values of the latent variable $Z$ given $x$. Similarly, conditional distribution $p_b(x|z)$ is referred to as decoder, since it decodes given latent feature $z$ into distribution of possible values of the random variable $X$ given $x$.

Parameters of this model $\phi, \theta$ are then calculated by minimizing the KL-divergence between $q_b(z|x)$ and $p_b(z|x)$, i.e. maximizing the following loss function:

$$\mathcal{L}(\theta; \phi; x) = -D_{KL}(q_b(z|x)||p_b(z|x))$$

By definition of a conditional distribution $p(z|x) = \frac{p(x, z)}{p(x)}$ we derive:

$$\mathcal{L}(\theta; \phi; x) = \mathbb{E}_{q_b(z|x)} \left[\log \frac{p_b(x|z)}{q_b(z|x)}\right]$$

$$= \mathbb{E}_{q_b(z|x)} \left[\log \frac{p_b(x|z)}{q_b(z|x)}\right] - \log p_b(x) \quad (13)$$

Since $p_b(x)$ is fixed, we only need to optimize the first term in Equation (14). By applying the definition of a conditional distribution again (i.e. $p(x, z) = p(x|z)p(z)$) we can further derive:

$$\mathcal{L}(\theta; \phi; x) = \mathbb{E}_{q_b(z|x)} \left[\log p_b(x|z) - \log q_b(z|x)\right]$$

$$= \mathbb{E}_{q_b(z|x)} \left[\log p_b(x|z) + \log \frac{p_b(x)}{q_b(z|x)}\right]$$

$$= \mathbb{E}_{q_b(z|x)} \left[\log p_b(x|z)\right] - D_{KL}(q_b(z|x)||p_b(z|x)) \quad (14)$$

The last formula in the Equation (15) is also referred to as evidence lower bound (ELBO) [12], [3]. Intuitively, the first term $\mathbb{E}_{q_b(z|x)}[\log p_b(x|z)]$ is the reconstruction error between the input to the encoder and the output of the decoder, the second term $D_{KL}(.)$ serves as a regularizer that measures the amount of information lost when $p(z)$ is used to approximate $q_b(z|x)$.

B. Autoencoder

We have used an implementation of both encoder $q_b(z|x)$ and decoder $p_b(x|z)$ based on neural networks. This type of neural network that learns to encode input $x$ into a latent representation $f(x)$ and to decode it back into approximation $g(f(x)) \approx x$ is called an autoencoder [11].

Each neuron $i$ in the neural network takes a $d$-dimensional input vector $x_i \in \mathbb{R}^d$ and maps it to a $d'$-dimensional output vector $y_i \in \mathbb{R}^{d'}$ with deterministic mapping (forward propagation):

$$y_i = s(W_i x_i + b_i) \quad (16)$$

where $s(.)$ is differentiable non-linear activation function, $W_i \in \mathbb{R}^{d \times d'}$ is a weight matrix and $b_i \in \mathbb{R}^{d'}$ is a bias. Model parameters $\theta = \{W, b\}$ are trained with Stochastic Gradient Descent (SGD) algorithm w.r.t. to loss function $\mathcal{L}$ by iteratively updating weight matrices and biases (backpropagation):

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta} \quad (17)$$

where $\eta$ is a learning rate that specify how much the parameters $\theta$ should be adjusted in each iteration of the SGD. In our implementation we have used an adaptive learning rate in which the gradient is divided by a running average of its recent magnitude, this approach is known as RMSprop [24].

The architecture of neural network used in our experiment is depicted in Figure 6. We have used Rectified Linear Unit (ReLU) as an activation function of neurons in hidden layers:

$$s_{ReLU}(x) = \max(0, x) \quad (18)$$

The neurons in output layer use sigmoid activation function, which ensures output normalization to $(0,1)$ interval as:

$$s_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (19)$$

C. Reparametrization trick

Optimizing the lower bound in Equation (15) with SGD is problematic, because the latent vector $z$ used in $p_b(z|x)$ must be sampled from distribution $q_b(z|x)$. However, the random sampling is a non-differentiable operation, hence it cannot be used for calculating gradients during backpropagation as in Equation (17). As shown in [16], the randomness of the sampling operation can be shifted into a random noise vector $\epsilon$ which allows us to express individual samples of reparametrized random variable $\tilde{z} \sim q_b(z|x)$ deterministically as:

$$\tilde{z} = \mu + \sigma \odot \epsilon \quad (20)$$

where $\epsilon$ is a random noise vector $\epsilon \sim \mathcal{N}(0, I)$ and operator $\odot$ is an element-wise product. This makes the model deterministic, which means that for the same parameters it always returns the same outputs, the only source of stochasticity is given by $\epsilon$. The differentiability of sampling using this reparametrization trick is illustrated in 5.
M can be set to 1 for sufficiently large number of minibatches. Hence only a randomly subset minibatch is used to approximate the expected value of the log-likelihood. The reconstruction error is then approximated by binary cross-entropy.

D. Variational autoencoder

As already mentioned, the distributions \( p(z) \) and \( q_\theta(z|x) \) are normally distributed:

\[
p(z) = \mathcal{N}(0, I) \\
q_\theta(z|x) = \mathcal{N}(\mu, \sigma^2 I)
\]

Using the closed form of the KL-divergence for two multivariate Gaussians gives us the following regularization term:

\[
D_{KL}(q_\theta(z|x)||p(z)) = \\
= \frac{1}{2} \sum_{j=1}^J \left( 1 + \log(\sigma_j^2) - (\mu_j)^2 - (\sigma_j)^2 \right)
\]

where \( J \) is the dimensionality of the latent vector \( z \) and the parameter distributions are mapped from a hidden layer \( h \) of the encoder \( q_\theta(x|z) \) (as depicted in Figure 6), i.e. \( \mu_j = W_h \mathbf{h} + \mathbf{b}_\mu \) and \( \log(\sigma_j)^2 = W_h \mathbf{h} + \mathbf{b}_\sigma \).

We assume that the decoder \( p_\theta(z|x) \) is Bernoulli distributed. The reconstruction error is then approximated by binary cross-entropy as:

\[
\mathbb{E}_{q_\theta(z|x)}[\log p_\theta(x|z)] \approx \\
\approx \frac{1}{L} \sum_{i=1}^L \left( \log y_i + (1 - x_i) \log(1 - y_i) \right)
\]

where \( x_i \) is the input of the encoder, \( y_i \) is the output of the decoder and \( L \) is the number of latent samples \( z_1, \ldots, z_L \) used to approximate the expected value of the log-likelihood.

The original dataset \( X \) consisting of \( N \) samples can be potentially very large. Hence only a randomly subset minibatch of \( M \) wafers \( X_M = \{x_1, \ldots, x_M\} \) is sampled from the whole dataset \( X \). The loss function for each minibatch is then:

\[
\mathcal{L}(\theta, \phi; X) \simeq \frac{N}{M} \sum_{i=1}^M \mathcal{L}(\theta, \phi; x_i)
\]

The number of sampled latent variables \( L \) in each iteration can be set to 1 for sufficiently large number of minibatches \( M \) [16] (in our implementation \( L = 1, M = 100 \)).

Fig. 5: Auto-encoding variational Bayes with reparametrization trick. Red line depicts the non-differentiable sampling operation. Instead of sampling \( z \) directly from \( q(z|x) \) as shown in (a), we encode the latent feature as \( z = \mu + \sigma \epsilon \) as shown in (b).

Fig. 6: Architecture of the variational autoencoder used in our experiments.

IV. CLUSTERING

In Section II, we have obtained a clean wafer dataset with comparable wafer measurements (data samples). In section III, the data samples have been modelled by approximating the probability distribution that generate these samples. We have described a mechanism for non-linear mapping of high-dimensional data samples \( x \) into a low-dimensional latent features \( z \) in a way that we can calculate the distance between different latent features. A two dimensional feature space trained on the wafer dataset is depicted in Figure 8. This allows us to efficiently find groups of similar samples in fully unsupervised manner.

Clustering (or cluster analysis) groups set of objects together based on a similarity between them. We typically recognize two types of clustering – hierarchical clustering which groups the set of objects into a hierarchical tree (dendrogram) and partitioning clustering which groups the set of objects into a disjoint subsets, such that each object is exactly in one subset.

Both clustering methods can be applied for our use case. The primary goal of this paper is to demonstrate capabilities of variational autoencoder on pre-processed wafer test data, hence we have selected only one representative from both types of clustering algorithms – \( k \)-means and Hierarchical agglomerative clustering. It is however possible, that other clustering methods may provide better clustering performance.

A. \( k \)-means

Given a set of \( n \) latent features \( \{x_1, x_2, \ldots, x_n\} \), we cluster them with \( k \)-means into a disjoint clusters \( C_1, C_2, \ldots, C_k \). The objective of this algorithm is to optimize squared Euclidean distance between the latent feature \( x \) and the centroid \( \mu_i \) of the cluster \( C_i \) till convergence:

\[
\arg \min_C \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2
\]

B. Hierarchical agglomerative clustering

This is a bottom-up approach to hierarchical clustering. Each sample starts in its own clusters and pairs of clusters are iteratively merged according to a distance measure between them. In our implementation we have used a Ward’s minimum variance criterion that minimizes the total within-cluster variance, i.e. clusters that increase the total variance the least are merged in each iteration. [1]
V. Evaluation

In order to evaluate the performance of the approach presented in this paper, we have compared the variational autoencoder approach with other commonly used decomposition methods. The same pre-processing and clustering algorithm have been used for the competing methods. Detected clusters for two dimensional latent space can be seen in Figure 7.

The Silhouette metric [21] was used to measure how similar a latent vector is to the other vectors in its own cluster compared to the other clusters:

\[
\text{sil}(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}
\]  

(26)

where \(a(i)\) is the average distance between vector \(x_i\) to the other vectors in the same clusters and \(b(i)\) is the average distance between vectors \(x_i\) to vectors in the nearest cluster. The Silhouette score ranges from -1 to 1. The higher this value is the better the clustering performance, values around 0 indicate overlapping clusters and negative values means that the sample was likely assigned to a wrong cluster. The average value of \(\text{sil}(i)\) over all latent features gives us results as shown in Figure 9.

References


Fig. 9: Evaluation of different feature extraction techniques with two clustering methods: \( k \)-means and agglomerative clustering. Approach based on variational autoencoder (VAE) yields better separated clusters (measured with Silhouette score) compared to the other methods in majority of cases.

2018 International Joint Conference on Neural Networks (IJCNN)