# WM-Graph: Graph-Based Approach for Wafermap Analytics

Min Jian Yang, Yueling (Jenny) Zeng, Li-C. Wang
*University of California, Santa Barbara*
*Santa Barbara, California 93106*

***Abstract*—This paper introduces WM-Graph, a novel approach designed for flexible analytics of wafermaps. The key concept behind WM-Graph is the construction of a wafermap graph, where individual wafermaps are connected if they exhibit semi-equivalence. This graph-based structure allows a wide range of analytics to be performed using established graph algorithms. Unlike traditional multi-class classification methods, WM-Graph enables more versatile analyses, making it possible to answer complex, practical questions that would otherwise be difficult to address. We explain the technical innovations that underpin the WM-Graph approach and demonstrate how to perform certain analytical tasks with simple graph operations. The effectiveness of the WM-Graph approach is validated through experiments using the public WM-811K dataset and a proprietary dataset from a recent production line.**

## 1. Introduction

A wafermap is an image showing passing and failing dies on a wafer. Wafermap analytics has been an important area of study in the semiconductor industry for decades. In the last decade, one specific problem, known as Wafermap Pattern Recognition (WMPR), became a focus of study after the publication of the TMSC WM-811K dataset [1] in 2015. This dataset includes 811,457 wafermaps among which 172,950 are labeled and the rest are not labeled. The work [1] introduced two ML-related approaches, one for classifying wafermaps and the other for searching similar wafermaps based on a given wafermap. The approaches were based on engineering a set of discriminative features to build a model for pattern recognition and similarity ranking.

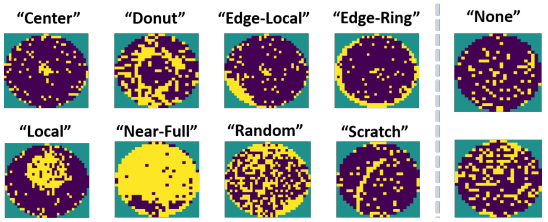


Figure 1. 8 patterns and "None" class defined in the WM-811K dataset

The work in [1] exemplifies treating wafermap analytics as solving a *multi-class classification problem*, a supervised learning problem in view of the general machine learning. The WM-811K dataset pre-defines nine classes. Figure 1 illustrates eight pattern classes where the purple color marks the area of a wafer, and the yellow pixels indicate the locations of failing dies on the wafer. In addition, a "None" class is used to classify wafermaps with "no pattern". The dataset includes different wafer sizes. One common size of wafermap image is $27 \times 25$ (with 518 dies).

### 1.1. Issues with pre-defined pattern classes

In our view, a multi-class wafermap classifier is not sufficient to support the practical needs of wafermap analytics. To illustrate this observation, Figure 2 shows two commonly-asked analytic questions to point out a gap between the questions and a wafermap classifier (as the tool).

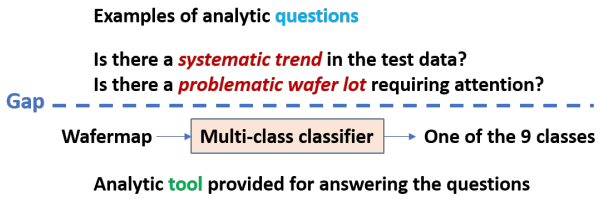


Figure 2. Gap between analytic questions and a classifier tool

Suppose the classifier is trained based on the WM-811K labeled dataset and let us put aside the accuracy issue with the classifier. The capability of the classifier is to label each wafermap with one of the nine classes. To answer the two questions shown in the figure, we need to first define the meaning of the terms "*systematic trend*" and "*problematic wafer lot*" based on the nine-class labeling capability.

Suppose we define systematic trend as the following:

```
Frequent occurrence of a pattern
within a given short time period
```

And accordingly we can define problematic lot as:

```
A lot with a systematic trend.
```

These definitions might sound reasonable at first. However, whether or not they make sense largely depends on the definition of the term "*pattern*" to begin with. And with WM-811K, patterns are limited in terms of the eight pattern classes shown in Figure 1.

To illustrate the issue of this limitation, Figure 3 below shows that wafermaps with the same class label might not be seen as having a "systematic trend". The figure shows two WM-811K pattern classes: "Edge-Local" and "Local". For each class, seven wafermaps are shown. In each class,

2378-2250/24/$31.00 ©2024 IEEE
DOI 10.1109/ITC51657.2024.00030



Regular Paper

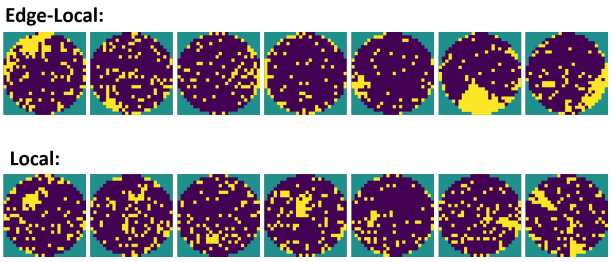


Figure 3. Within-class pattern variations seen in WM-811K dataset

whether or not those seven wafermaps can be considered as having a systematic trend is questionable.

When the term "systematic trend" is used in an analytic question, it is likely that one is actually interested in observing a "systematic issue". Then, the underlying question for whether or not to treat the seven wafermaps in Figure 3 as a systematic trend depends on one's belief if they are due to the same manufacturing issue.

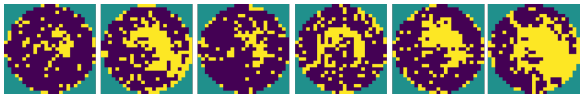

Figure 4. Patterns vary within a single lot in WM-811K dataset

To further illustrate the issue, Figure 4 shows six consecutive wafermaps from one single lot in the WM-811K unlabeled dataset. These six wafers visually show a systematic trend. However, according to WM-811K's class definition, the 1st, 3rd, and 5th wafers (from the left) are "Local". The 2nd and the 4th are "Donut". The rightmost one can be classified as a "Center" or "Local" (because of label ambiguity [2]). From the WM-811K pattern class definition, these wafermaps are separated into three classes and hence, they are considered as different. Nevertheless, by the fact that they are consecutive wafers within the same lot, calling them a systematic trend is very much justifiable.

### 1.2. Three postulates for the work

When a wafermap analytic tool is a classifier, this classifier is built upon a dataset like the WM-811K [1] with a fixed definition of pattern classes *in advance*. This pattern class definition might not be sufficient in view of a future application context. With a fixed class definition, it is assumed that in an application, there is no need to differentiate wafermaps beyond those defined classes. This might not be true, as pointed out in the discussion above.

In fact, the term "systematic trend" can depend on the data itself and given an analytic context, the term might have a dynamic meaning. For example, in early stage of a production when yield is low, the emphasis might be on the *persistence* of a trend to impact the yield. In later stage when yield is high, the emphasis might be on capturing an *excursion of the yield*, e.g. see the work in [3].

In different analytic contexts, the most suitable pattern class definitions can be different. The WM-811K provides one particular pattern class definition, which can be rather limited in view of a particular analytic context. In general, a multi-class classifier fixed upon one particular pattern class definition for one analytic context can be too limited to be applied in another context.

Providing a pattern class definition in advance and expect that the definition will make sense universally to all analytic contexts in the future, can be a rather unrealistic assumption. Consequently, we desire an analytic tool that can enable *flexible pattern class definition* which depends on both the data and the context. We use three postulates below to highlight this thinking, which is fundamentally different from the multi-class classification approach:

1) Pattern class definition should be flexible enough to support answering various analytic questions.
2) Pattern class definition can depend on the dataset associated with the analytic question.
3) A user can assert a pattern class, e.g. by giving at minimal one wafermap as an example.

In this work, we will present a wafermap analytic tool that can provide capabilities in view of the three postulates. The tool is built upon the approach called WM-Graph (WaferMap Graph) incorporating two essential ideas: (1) Build a wafermap *recognition graph* where *semi-equivalent* wafermaps are connected and (2) Enable all required analytic capabilities through graph-based operations. Note that the term "semi-equivalent" will become clear when we discuss the method to construct a recognition graph.

The rest of the paper is organized as the following. Section 2 reviews other works done on the WM-811K dataset. Section 3 explains our *question-dataset pair view* and how to use *concepts* to phrase an analytic question. Section 4 lists a number of graph-based concepts for interpreting the various analytic questions presented in Section 3. Section 5 explains the MINIONs approach first proposed in [2] and how it can be the foundation to realize the proposed WM-Graph approach. Section 6 then presents detail of our current MINIONs implementation. Section 7 uses experiment results to demonstrate how analytic questions mentioned in Section 5 can be answered in practice. Section 8 concludes.

## 2. Literature review

A number of works has been published using the WM-811K dataset. These works can be roughly divided into two categories: (1) those following a traditional machine learning (ML) approach, where a set of *features* are developed and used to convert each wafermap image into a *feature vector*, followed by applying a *model building method* that operates on the set of feature vectors to learn a model; (2) those following a *deep learning* (DL) approach, where the feature extraction step is automated by, for example, layers of a Convolution Neural Network (CNN), followed by one or more Fully-Connected (FC) layers for classification decision (corresponding to the model building step in the traditional

ML). Figure 5 summarizes these two common approaches for training a wafermap classifier.

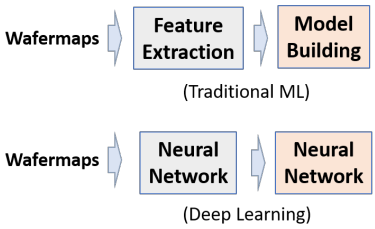


Figure 5. Two common ML approaches for training a classifier

Early works from 2015 to 2017 followed the traditional ML approach. The original work [1] belongs to this category. Then, the work [4] adopted a slightly different feature set and aimed to improve the multi-pattern detection accuracy, i.e. capturing a wafermap with two or more predefined pattern classess. The work [5] advocated using more discriminant features based on Linear Discriminant Analysis (LDA) to simplify the model building step. Using Radon transform features, the work [6] proposed a special Decision Tree based ensemble learning method to improve interpretability of the model.

From 2018, applications of DL on the WM-811K dataset began to emerge, e.g. using a CNN architecture for training the classifier. The author in [7] proposed a 2-stage classification: first between pattern and no pattern (i.e. to filter out the "None" class) and then, for those containing a pattern, classify which class it belongs to. WM-811K is an extremely imbalanced dataset where some classes have many more samples than others [1]. The work in [8] therefore proposed a data augmentation method based on Generative Adversarial Network (GAN) [9]. Instead of using GAN, the authors in [10] used Auto Encoder (AE) [11] for data augmentation. In contrast, the authors in [12] used predetermined rules to augment the dataset.

The authors in [13] approached the problem from a different angle, by addressing the concern that a wafermap to be predicted might contain a new pattern or a multi-pattern not defined with the dataset. Therefore, the work proposed using *selective learning* to determine applicability of a model, where the model included the choice to abstain from making a prediction, i.e. the model could reject some samples and only make prediction on others.

Recently, the work in [14] proposed a semi-supervised learning approach, attempting to reduce the number of labeled samples required for the training. Similar to previous works, a set of transformations was identified for data augmentation and used to adjust the imbalanced data. Their result showed a tradeoff between the prediction accuracy and the amount of training samples in use.

For all the works reviewed above, it suffices to say that they start with a multi-class classification problem defined with the WM-811K dataset. Our work takes a fundamentally different path and does not start with a defined classification problem. In our problem formulation, defining pattern classes *is part of the problem.*

# 3. Question-Dataset pair view


Figure 6. Question-Dataset pair view with the WM-Graph approach

Figure 6 illustrates the high-level problem formulation seen by WM-Graph. Input to the tool is a pair of analytic question and dataset. We can assume the dataset is a set of wafermaps without class label. Output is given as a plot, based on the selected subset of wafermaps. The question consists of two parts: *concept* and *constraint*. Essentially, we can think of WM-Graph as a filtering-in process that selects wafermaps based on the concept and constraint specified in the question. Depending on the question, labels can be associated with the selected wafermaps as part of the output.

## 3.1. Analytic question

In this work, we consider analytic question stated with a concept $C$ and a constraint $S$. Later we will explain $C$ and $S$ in detail. For now, we can simply think of $C$ and $S$ as two types of *wafermap filters* used to select wafermaps. Note that $S$ can be null, i.e. there is no constraint. With $C$ and $S$, we consider two types of questions:

**Existence type** The general form of this type can be like: "Is there $C$ under the constraint $S$?" The following provides a few examples where the concept name is capitalized and constraint is in italic.

Is there a PATTERN *with yield loss effect* $\geq 0.5\%$?
Is there a SYSTEMATIC PATTERN?
Is there a PROBLEMATIC LOT?
Is there a SYSTEMATIC TREND *last month?*

**Search type** The general form of this type can be like: "Given $C$, does $C$ appear in the dataset constrained by $S$?" The $C$ is provided through an example. For example, the $C$ can be specified with one particular wafermap (or multiple wafermaps) or with one particular wafer lot. The following provides two example questions of this type.

Given wafermap $wp$, are there other wafermaps *similar* to $wp$?
Given wafer lot $L$, are there other *similar* lots?

Note that the answer for a search type of question depends on how WM-Graph interprets the word "*similar*".

## 3.2. Concept

As shown above, a *concept* has a name, e.g. a word or a phrase. Regardless how a concept is called, the important aspect is that a concept must be associated with a *decision procedure* that decides its existence in a given dataset. Specifically, we define concept as the following.

*A concept is something whose existence in a dataset is decidable by a software script.*

In our implementation, a software script is a Python script. In the context of wafermap analytics, the decision procedure can be seen as a filter to select wafermaps from a given set of wafermaps.

### 3.3. Constraint

We can consider four types of constraints: yield constraint, time constraint, test constraint, and lot constraint. A constraint is used to limit the scope of the data to be applied with the concept. For example, "yield loss $\geq 0.5$" is a yield constraint. "January to April" is a time constraint. "Test bin X" is a test constraint. A wafer lot constraint can be given by explicitly specifying the names for a set of lots.

## 4. Graph-based concepts

To process an analytic question, the most important step is to decide on the meaning of the concept mentioned in the question. Constraints are more specific and their interpretation is usually without ambiguity. In this section, we will present a list of concepts built upon a so-called MINIONs graph of wafermaps [2]. More detail about MINIONs graph will be discussed in Section 5. In this section, we focus on the properties of a MINIONs graph.

### 4.1. The semi-equivalence relation $\approxeq$

A MINIONs graph $M_G$ is constructed based on a given set of $n$ wafermaps $wp_1, wp_2, \ldots, wp_n$. Each wafermap $wp_i$ corresponds to a node in $M_G$. Two wafermaps $wp_i, wp_j$ connected with a bidirectional edge in $M_G$ iff the relation $wp_i \approxeq wp_j$ holds.

The binary relation "$\approxeq$" is called *semi-equivalent*. This relation "$\approxeq$" is the foundation for building a MINIONs graph and consequently, all the concepts defined with WM-Graph are based upon it. Implementation detail for "$\approxeq$" will be provided in Sections 5 and 6.

### 4.2. Other relations built upon $\approxeq$

Once we have the semi-equivalence relation $\approxeq$ established, it can be used to define other relations.

**Equivalent** denoted as "$\equiv$": Given a group of wafermaps $E = \{wp_1, \ldots, wp_k\}$, for $k \geq 3$, we say that they are equivalent if $\forall i, j,\ 1 \leq i, j \leq k$, we have $wp_i \approxeq wp_j$. In this case, we say that $wp_i \equiv wp_j\ \forall wp_i, wp_j$ in the group.

Intuitively, this definition is to say that on the MINIONs graph, the group $E$ of wafermaps forms a *clique* of size at least 3. If a group of wafermaps form a clique of size $\geq 3$, then we consider them all equivalent.

**$i$-similar** denoted as "$\precsim_i$": We say that two wafermaps $wp_a, wp_b$ are $i$-similar, denoted as $wp_a \precsim_i wp_b$ if the following holds: There exists no wafermaps in between (let $\precsim_0$ be the same as $\approxeq$), or a sequence of wafermaps $w_1, \ldots, w_i$, $i > 0$ such that $w_a \approxeq w_1 \approxeq w_2 \approxeq \cdots \approxeq w_i \approxeq w_b$.

**Distance** The smallest $i$ where $wp_a \precsim_i wp_b$ is called the *distance* between $wp_a$ and $wp_b$.

With the wafer-to-wafer distance defined above, we can also define the wafer-to-group distance such that for a wafermap $wp$ and a group of wafermaps, the distance from $wp$ to the group is the smallest of the wafer-to-wafer distances to all individual wafermaps in the group.

### 4.3. Support concepts

With the relations defined above, we can then define a number of *support concepts*. These concepts are supporting in nature as they can be referred directly in a given analytic question, or they can be used to construct other higher-level concepts referred in a given analytic question.

First, we define four types of pattern concepts.

**Anchor Pattern:** Given a set of wafermaps, an anchor pattern is represented by a maximal clique in the MINIONs graph, i.e. a maximal group of mutually equivalent wafermaps $A = \{wp_1, \ldots, wp_k\}$ such that there exists no $wp \notin A$ that $wp$ and $wp_i$ are semi-equivalent $\forall wp_i \in A$.

**Primitive Pattern:** A primitive pattern is represented by a group of anchor patterns $A_1, \ldots, A_l$ for $l \geq 1$ such that for any pair $A_i, A_j$, there exists a waferamp $wp$ such that $wp \in A_i$ and $wp \in A_j$. Intuitively, on the MINIONs graph the two maximal cliques $A_i, A_j$ share one or more nodes. In this case, $A_i, A_j$ are considered the same primitive pattern.

**$i$-Hop Pattern:** An $i$-hop pattern is based on a primitive pattern and includes all wafermaps that has a distance $i$ to any wafermap in the primitive pattern wafermap set. Intuitively, a primitive pattern on a MINIONs graph is represented by a set of nodes. An $i$-hop pattern is an expansion from this set to include all nodes reachable within distance $i$.

**Lot-Pattern:** A lot-pattern is a collection of primitive patterns appearing in the same lot, i.e. they are considered as a single pattern and may be given with a unique pattern name.

With the four pattern concepts defined above, other pattern concepts can be defined based on them, depending on the need to interpret an analytic question. Below we will show a few examples. It can be seen that with the support concepts, one can conveniently define more complex concepts to support answering various analytic questions.

### 4.4. Systematic Trend

A systematic trend can be defined with a window of consecutive wafer lots $S = \{L_1, \ldots, L_m\}$. We say that $S$ contains a systematic trend iff for any lot in $S$, it contains a pattern that also exists in at least one other lot. The definition of this "pattern" can be chosen as the anchor pattern, the primitive pattern, the $i$-hop pattern, or a more complex pattern defined based on the supporting pattern concepts.

To give an example, let us say $P_1, P_2, P_3, P_4$ are four pattern concepts. Let us say $L_1 \supseteq \{P_1, P_2\}$, i.e. $L_1$ contains the patterns $P_1, P_2$, $L_2 \supseteq \{P_2, P_3\}$ and $L_3 \supseteq \{P_3, P_4\}$. Then,

a window containing $\{L_1, L_2, L_3\}$ has a systematic trend because $L_1, L_2$ share the pattern $P_2$ and $L_2, L_3$ share the pattern $P_3$. Notice that a systematic trend can contain a morphism of the pattern across a sequence of wafer lots.

As seen, with our WM-Graph approach the concept of "systematic trend" can be flexibly defined. This flexibility is a benefit provided by our approach, which is hard to accomplish with the traditional multi-class classification approach.

### 4.5. Problematic lot

With our WM-Graph approach, we can also provide flexibility to define what constitute a problematic lot. The term "problematic" is defined in terms of two aspects: yield loss and *systematic strength*. The yield loss for a given lot can be calculated directly. Hence, our focus is on how to define the term "systematic strength".

**Systematic Strength:** Given a group of wafermaps $wp_1, \ldots, wp_n$, we measure a *systematic strength* for the group based on the pairwise distances between wafermaps, where the *distance* is defined in Section 4.2 above. Given two wafermaps $wp_i, wp_j$, their distance is denoted as $dist(wp_i, wp_j)$. The systematic strength for the group is calculated with the following formula:

$$\frac{1}{n}\Sigma_{\forall i,j}\frac{1}{(dist(wp_i, wp_j)+1)^2} \quad (1)$$

Note that according to our distance definition above, if $wp_i, wp_j$ are directly connected with an edge in the MINIONs graph, then $dist(wp_i, wp_j) = 0$.

Intuitively the systematic strength tries to measure an average similarity across all wafermaps in the group. The higher this average similarity value is, the more likely the group has a problem. This is to contrast to the situation that if the group does not have a problem, then those wafermaps should appear to be random.

**Systematic Strength w.r.t $wp$:** The systematic strength for a group of wafermaps can also be measured with respect to (w.r.t) a given wafermap $wp$. This can become handy when answering a search type of question (see Section 7 for its use). To define this measure, we modify formula (1) above into the following ($wp_i$ is from the group):

$$\Sigma_{\forall i}\frac{1}{(dist(wp, wp_i)+1)^2} \quad (2)$$

**Problematic Lot:** With systematic strength defined, we can then say that a problematic lot is a lot with a high yield loss *and* with a large systematic strength. Therefore, to inspect a large number of lots we can use a two dimensional plot to find those problematic lots (see Section 7 for detail).

## 5. The MINIONs approach

The previous section discusses the various ideas in our WM-Graph approach for supporting wafermap analytics. As seen, the foundation to accomplish all those ideas starts with a MINIONs graph built upon a given set of wafermaps. The essential capability to enable this graph construction is to implement the semi-equivalence relation $\cong$. To be clear, the following states the requirement fundamental to our WM-Graph approach:

> *A decision procedure that given two wafermaps decides if they can be treated as the same or not.*

### 5.1. Concept recognizer for one wafermap

In the earlier works [15][2], this decision procedure is called a *concept recognizer*. Given a wafermap $wp$, $wp$ is treated as representing a (mini) concept on its own and a recognizer is built to decide the existence of this concept. Training a concept recognizer based on just one wafermap [15], is called MINIONs which stands for MINiture Interactive Offset Networks [2]. The MINIONs approach explained in this section is based on a training scheme (discussed in Section 6) improved from that in the previous work [2].

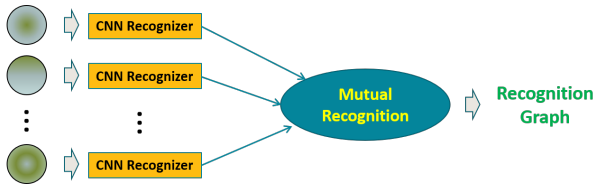


Figure 7. High-level idea of the MINIONs approach

Figure 7 illustrates the high-level idea of the MINIONs approach. A neural network (NN) model is independently learned for each wafermap. This NN model, called a MINION, serves as the concept recognizer specific for the wafermap. With one MINION for every wafermap, we can then perform *mutual recognition* on pairs of wafermaps, which will result in a *recognition graph*. In this graph, every node is a wafermap. Two nodes have an edge connecting them if they are mutually recognized. With MINIONs, the mutual recognition relation is therefore treated as the semi-equivalence relation $\cong$ defined earlier.

### 5.2. MINIONs' one-shot learning

A MINION is trained with one sample. Training with one sample is generally referred to as *one-shot learning* [16][17]. In machine learning, three general approaches had been proposed to tackle one-shot learning: data augmentation, transfer learning, and meta learning (see their discussion in view of training a MINION model in [18]). For training a MINION, the earlier studies [2][18] did not find these approaches effective. A fourth approach called *manifestation learning* was adopted [18].

Figure 8 explains the idea of manifestation learning presented in [15] and further improved in this work with a more effective training scheme. The training has two phases. In the preparation phase, popular hand-writing digit classification dataset MNIST [19] is used. The MNIST dataset

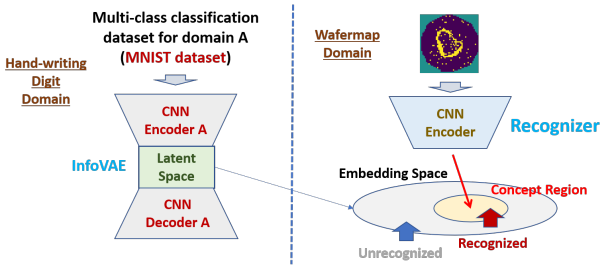


Figure 8. The idea of manifestation learning presented in [15]

contains 5000 images for each digit, 0 to 9. In manifestation learning, this dataset is used first to train a Variational Auto-Encoder (VAE) model [20]. A VAE model comprises a CNN encoder and a CNN decoder. The CNN encoder maps each MNIST image onto a distribution of *embedding vectors* in the *latent space*. Then, the CNN decoder maps an embedding vector back to an image. In [15], this training is based on InfoVAE [21].

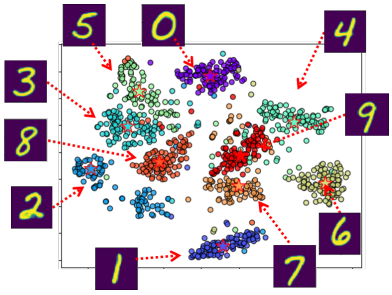


Figure 9. Visualization of latent space learned with MNIST samples

In our implementation, the encoder transforms an input image into a 16-dimensional embedding vector in the latent space. Figure 9 illustrates how this latent space looks like by projecting 16-dimensional vectors onto 2-dimensional vectors using TSNE [22]. This illustration uses only 100 samples for each digit.

After the InfoVAE training to obtain the latent space with embedding vectors, on its latent space we build an SVM one-class model [23] to capture a so-called *concept region* for a selected class. The selected class used in [2] was digit "1". The SVM model is built in such a way that it tries to capture as many embedded vectors of digit "1" as possible and as few vectors from any other digits as possible, i.e. it is a conservative model.

Refer to Figure 8. In the one-shot learning phase, the training takes place in the wafermap domain to train an encoder without the decoder. The latent space with embedded vectors from the MNIST training is *transferred* to serve as the target for the wafermap encoder. The same SVM model is reused for training a recognizer for every wafermap. The idea in this training is to map the wafermap to the center of the concept region defined by the SVM model.

After the training, the CNN encoder is our recognizer (MINION) for the given wafermap. Then, for a wafermap given as input to the recognizer, it will be mapped to an embedded vector. If this vector falls inside the concept region as determined by the SVM model, it is recognized. Otherwise, it is unrecognized.

## 6. Detail for training a MINION

While the early works described some of the MINION's training details [15][2], our current implementation has been enhanced over time with the development of IEA-PLOT reported in [24]. Here, we consolidate the most updated training setup of the MINION models as follows.

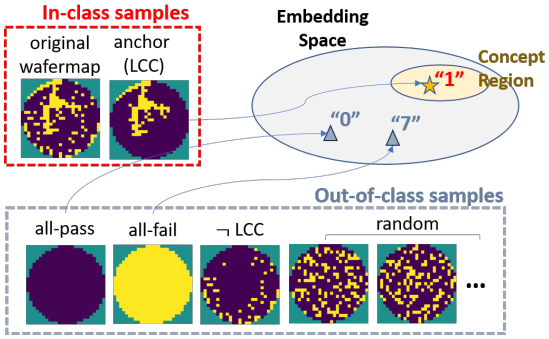


Figure 10. Detail of training a MINION model

As mentioned above, the learned latent space along with the SVM concept region model are transferred to be used in the MINION's one-shot training. Figure 10 shows the training data used in the MINION training, including the original wafermap and some augmented images.

A MINION is trained by mapping the *anchor* image onto the latent vector with the highest SVM score among all the "1" samples inside the concept region. Denote this latent vector as $t_1$. This anchor image is a transformation of the original wafermap, by extracting the *largest connected component* (LCC) from the original wafermap. A connected component (CC) is defined based on the 8-neighbor die connectivity, i.e. if one failing die (pixel with value 1) is the neighbor in any of the eight directions of another failing die, the two dies are considered "connected". A CC is a set of failing dies that are connected. The LCC is the largest CC in terms of the number of failing dies connected.

In addition to the anchor, the *in-class* set contains the original image. There are also *out-of-class* training samples used in the training. The out-of-class set includes an "all-pass" wafermap, an "all-fail" wafermap, a special wafermap by complementing the anchor image, denoted as "¬LCC", and wafermaps randomly generated with the count of failing dies comparable to that of the original wafermap.

All training data are resized to 64× 64 images. Figure 11 shows the neural network layers used in the MINION model. The input image is encoded by two CNN layers along with a Max Pooling layer, and then transformed into a 16-dimensional *embedding vector* $v$ by a dense layer. In addition, the SVM score of $v$ with respect to the latent space can be calculated by applying the scoring function

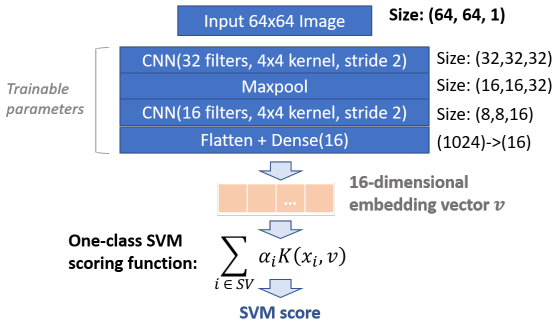


Figure 11. The neural network architecture of a MINION model.

of the one-class SVM model. In the scoring function, $K$ is the RBF kernel, $x_i$ is a support vector, $\alpha_i$ is the coefficient of support vectors [23].

It is interesting to note that in an early implementation, the in-class training set included additional images obtained by rotating the original wafermap. While this strategy might help a MINION recognize a rotated version of original wafermap, it made the MINION model less accurate to recognize the original wafermap. As a result, we abandoned this strategy and used only the two in-class images as described above. After a MINION is trained, when performing actual recognition, the input wafermap is used to generate multiple inputs by rotating with $\pm x°$, e.g. $\pm 10°$. Then, if any of the inputs is recognized, we consider the original input as recognized. In this way, a MINION is still able to recognize a rotated version of the original wafermap.

### 6.1. Triplet loss training with dual objectives

As shown in Figure 10, training a MINION uses various kinds of images. The in-class includes two images: the anchor (let it be denoted as "AN") and the original wafermap (denoted as "ORI"). The out-out-class ("OUT-C") includes the all-pass ("AP"), the all-fail ("AF"), the ¬LCC, and random images.

The high-level training objective for the MINION aims at aligning the output embedding vectors based on the concept region defined by the SVM model. Specifically, we desire the embedding vector of the in-class image "ORI" be inside the concept region, while the embedding vectors of the out-of-class images are positioned away from the concept region. On top of both, we desire the anchor "AN" to be mapped to the embedding vector with the highest SVM score. To achieve all these objectives, the loss function contains two parts: *vector mapping* and *triplet loss* [25].

To achieve vector mapping, we use a loss equation as below. Recall that $t_1$ is the vector with the highest SVM score. We let $f$ represent the mapping function of the neural network in the current training iteration.

$$\mathcal{L}_{vector} = ||f(\text{AN}) - t_1||^2 + ||f(\text{AP}) - t_0||^2 + ||f(\text{AF}) - t_7||^2 \quad (3)$$

The $t_0$ is the embedded vector that gives the median SVM score among all the vectors corresponding to digit "0" images. Similarly, the $t_7$ corresponds to the digit "7" image. We chose digits "0" and "7" because their embedded vectors in the latent space form a distribution that is the two farthest from the distribution of digit "1" vectors.

The second part of the loss function is a variant of the Triplet Loss [25], and is shown below.

$$\mathcal{L}_{tri} = max(T_{in} - \phi(\text{ORI}), 0) + max(\phi(\text{OUT-C}) - T_{out}), 0) \quad (4)$$

The $\phi()$ represents the SVM scoring function. The "OUT-C" is a randomly-selected out-of-class image each time. $T_{in}$ is a constant calculated as $T_{in} = \phi(t_1) - 0.05$, i.e. we desire the image "ORI" to be mapped to some vector whose SVM score is within 0.05 from the highest SVM score (the score of $t_1$). Moreover, $T_{out}$ is a constant calculated as $T_{out} = \phi(t_0) + 0.2$, we desire the out-of-class image to be mapped to some SVM score no greater than $T_{out}$. This is because the score $\phi(t_0)$ is rather small and hence, an adjustment is needed to relax the objective to make it more attainable.

The final loss is the sum of $\mathcal{L}_{vector}$ and $\mathcal{L}_{tri}$ and in the training, this loss is minimized. During training, the standard mini-batch stochastic optimization is employed. Specifically, an Adam optimizer [26] is used with a learning rate of 0.001 and batch size of 10. Model selection is based on selecting the model that obtains a loss value no more than 0.02 and has the lowest loss value among all models trained over a total of 300 epochs.

In addition to the hyperparameter selections typically involved in standard neural network training, MINION's training includes consideration of several other important parameter choices. For example, the number of augmented out-of-class samples is a user-specified parameter. The image transformation of a wafermap can involve more complicated settings. The random wafermap generation can be based on a fixed or a range of yield values. The heuristic used to extract the LCC wafermap involves several parameters such as the size of the CC and the number of CCs to retain, etc. The parameters used in our MINION training were determined empirically through extensive experiments, based on performance observed on the public WM-811K dataset and private test datasets from three production lines.

## 7. Experiments - Answering analytic questions

One of the most important objectives in wafermap analytics is to detect a yield issue, collect evidence, and generate an action. Analyzing "patterns" is just a means to attain this objective. An analysis has to start from somewhere. In view of the objective, perhaps the most relevant question is:

**Is there a SYSTEMATIC TREND in the dataset?**

The key to answer this question is to define the concept "SYSTEMATIC TREND". Based on the definition of systematic trend described in Section 4.4 and the definition of primitive pattern described in Section 4.3, Figure 12 shows an answer with two trends. The primitive patterns are named as "P##", shown below each wafermap. The dataset had

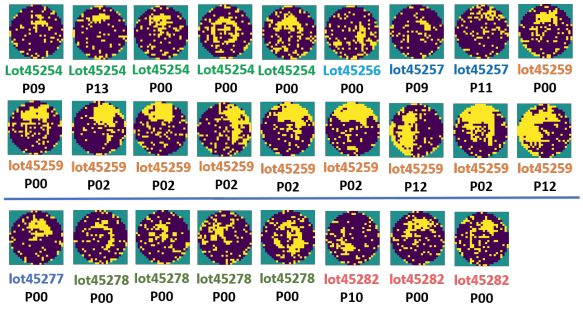


Figure 12. Two systematic trends found on WM-811K wafermaps

7038 wafermaps in 306 lots from WM-811K, based on the wafer size $27 \times 25$ (518 dies). There were 14 trends found in this dataset, each with four or more wafers across at least two lots. Figure 12 shows two examples to demonstrate the capability of using WM-Graph to answer this question.

Another relevant question in view of the objective is:

**Is there a PROBLEMATIC LOT in the dataset?**

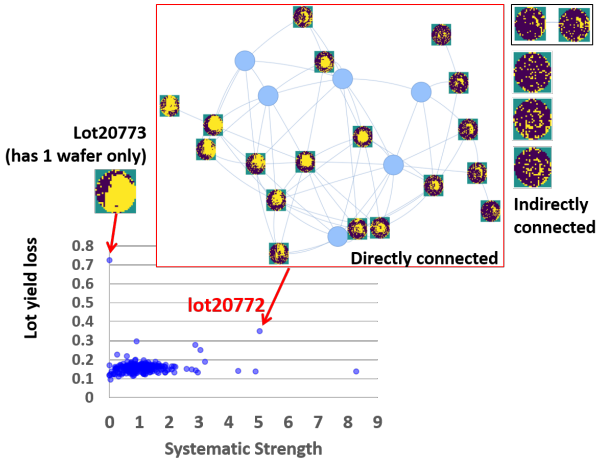


Figure 13. From the left plot, pick the problematic lot 20772 to show

Again, the key is to interpret the term "PROBLEMATIC LOT" and our interpretation has been discussed before in Section 4.5. That is, a problematic lot has high yield loss and large systematic strength. Figure 13 shows a 2D plot where x-axis is the systematic strength and y-axis is the lot-based yield loss. From the plot, we can pick "lot20772" as the problematic lot and its wafermaps are shown with a subgraph extracted from the MINIONs graph. On this subgraph, wafermaps from the lot are directly connected with at least another wafermap in the lot. Visually, we can see a trend formed by those wafermaps. There are also five other wafermaps (shown outside the red box) which are indirectly connected to the wafermaps in the subgraph.

As an interesting side note, on the plot shown in Figure 13 the highest yield loss is from "lot20773". This lot has only one wafer and the wafermap is shown.

Suppose now we decide that lot20772 is a problematic lot. Based on this lot, we desire to find all other lots showing the same problem, by asking the question:

**Other lots showing the SAME PROBLEM as lot20772?**

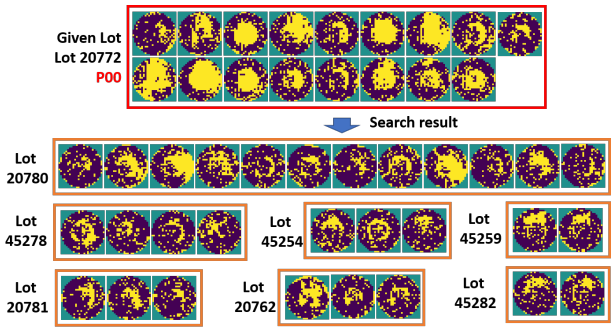


Figure 14. Lot-based same-problem search result for lot20772

As shown in Figure 14, lot20772 contains only one primitive pattern that is named "P00". The answer to this question can be based on finding all other lots containing the primitive pattern P00. There were 22 lots found in the search. Figure 14 shows 7 of them, where the number of P00 wafermaps in each lot is equal to or greater than two.

Answering the above three questions is based on finding the primitive patterns on the MINIONs graph. The "SYSTEMATIC TREND" and "PROBLEMATIC LOT" are two convenient concepts to start the investigation. In some cases, one might be interested in examining the data from the "PATTERN" perspective to check if there are other issues not showing up as a "SYSTEMATIC TREND" or as a "PROBLEMATIC LOT". In this case, we can ask the following questions to check out other potential issues.

**What PATTERNs are in the dataset?**

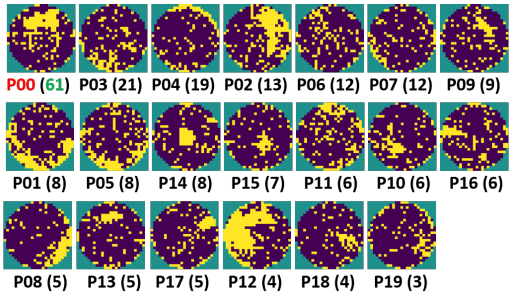


Figure 15. 20 primitive patterns found, named "P00" to "P19". In each case, the # of wafermaps in the pattern set is shown, e.g. P00 has 61 wafermaps.

According to the support concepts defined in Section 4.3, a primitive pattern is a set of connected maximal cliques in the MINIONs graph. Given a graph, there can be an enormous number of maximal cliques. While it is possible to enumerate all the maximal cliques as a start, this might not be the most effective way. One alternative is to start with a set of rules to narrow the selection to those maximal cliques satisfying certain intuitive properties.

With this idea, we manually wrote 8 rules. These rules were based on features commonly used for analyzing

wafermaps (see, e.g. [1]) These rules were used to describe our intuitive notions about the following 8 patterns: center, donut, arc, cluster, ring, catastrophic, grid, and scratch. These rules were conservative rules to ensure that they captured the "obvious" cases satisfying our pattern notions. Using the rules, we collected a set of wafermaps each satisfying at least one rule, as the starting point. For each wafermap in the set, we found the anchor pattern(s) containing the wafermap. If there was none, then the wafermap was simply ignored. Then, extending from the anchor patterns we established the primitive patterns.

We omit discussion of the detail for the rules, as they are not that crucial. They are auxiliary to provide a starting point. Keep in mind that with WM-Graph, the important question is never about the accuracy of a pattern classification scheme. Rather, when we obtain a set of primitive patterns, the important question is whether or not they are sufficient for us to attain the analytic objective, that is, to detect a yield issue, collect evidence, and generate an action. It is important to note that regardless of the rules, the primitive patterns have to satisfy the graph properties as defined in Section 4.3. In this sense, rules are used as a guide to direct our attention to certain patterns but not as a definition for a pattern. The pattern definitions are always graph-based in WM-Graph.

Figure 15 shows that there were 20 primitive patterns found, ordered by the number of wafermaps in each pattern set. For each pattern, one representative wafermap is shown. This wafermap is the one with the largest LCC size from the largest maximal clique in the primitive pattern set.

Suppose one find a wafermap $wp$ and is concerned with the issue shown on the wafermap. In this case, the following question might be asked:

**Is there a lot having the SAME ISSUE as $wp$?**

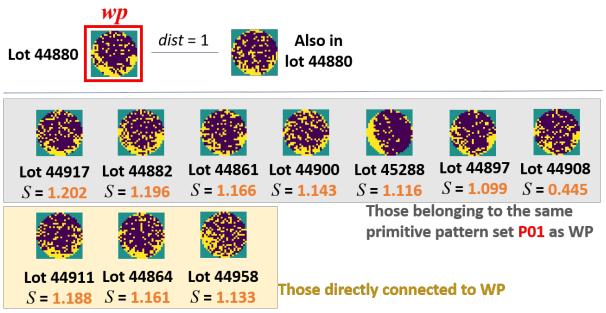


Figure 16. Ten lots each containing one wafermap having the "same issue"

Figure 16 shows the result for a selected $wp$ from lot 44880. First, the $wp$ is found to be in the primitive pattern set P01. There are seven other wafermaps in P01, each belonging to a different lot. Then, there are three other wafermaps directly connected to $wp$. These three also belong to three different lots. For each lot, we calculate the lot's "systematic strength w.r.t. $wp$" using the formula (2) in Section 4.5 and denote this as $S$ in the figure. As seen, the highest $S$ is with lot44917 and is only 1.202, which means that there is only one wafermap "close to" $wp$. This is indeed the case when we checked on other wafermaps in lot44917. The next closest wafermap had a distance 6 to $wp$. Because all other lots have a smaller $S$ than lot44917, on all those lots, the "issue" also affects only one wafer.

**What PATTERNs are in the dataset?**

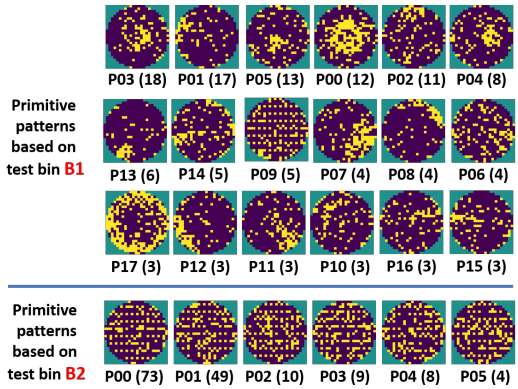


Figure 17. Similar results as that shown in Figure 15 before

Next, we will show results based on the private dataset containing 12782 wafers in 666 lots. Figure 17 shows the primitive patterns found in two separate analyses, each focusing on fails from a particular test bin (bin B1 and bin B2). Notice that the six primitive patterns based on B2 all show a "grid" with some minor variations, i.e. from the perspective of MINIONs graph, they are not the same.

**Is there a SYSTEMATIC TREND in the dataset?**

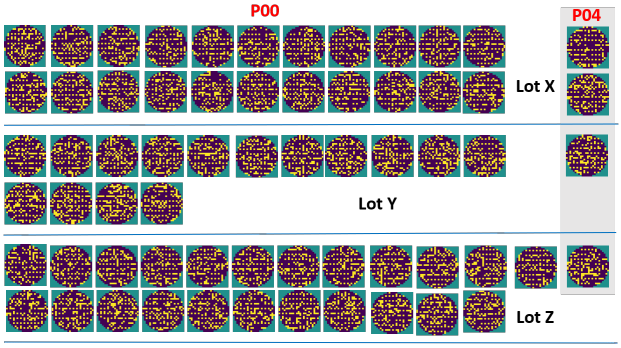


Figure 18. Similar results as that shown in Figure 12 before

Then, asking the systematic trend question based on bin B2 found many trends with the grid patterns. Figure 18 shows one of the grid trends based on three lots, named X,Y,Z. In these lots, most wafermaps belong to B2's P00 pattern with a few to B2's P04 pattern.

**Is there a PROBLEMATIC LOT in the dataset?**

Figure 19 shows the result by asking the problematic lot question. Wafermaps in four selected lots are shown, including the one with the highest yield loss and the one with the largest systematic strength. The two lots highlighted

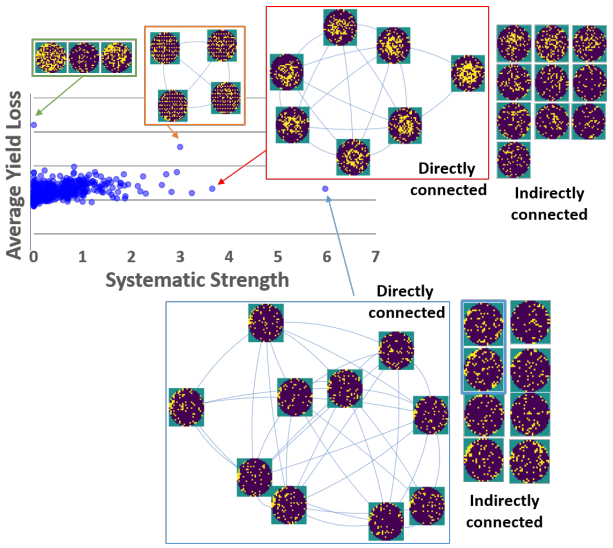


Figure 19. Similar results as that shown in Figure 13 before

with red box are more interesting, as one showing a grid-like pattern and the other showing a donut pattern.

## 8. Conclusion

This paper introduces WM-Graph, a graph-based framework designed to enable flexible wafermap analytics. WM-Graph can support a comprehensive set of graph-based concepts and operations, allowing developers to build higher-level constructs to address complex analytical questions, such as identifying systematic trends or detecting problematic lots. Experimental results are provided to showcase some of the WM-Graph's functionalities. WM-Graph facilitates wafermap analytics at a conceptual level, allowing concepts to be described using natural language phrases. This capability is essential for the development of the AI assistant, IEA-Plot, first introduced in [24].

**Acknowledgment** This work is supported in part by National Science Foundation Grant No. 2006739. The authors are thankful to Sergio Mier, Leon Wang and Patty Pun of Qualcomm for their valuable inputs to our research.

## References

[1] M.-J. Wu, J.-S. R. Jang, and J.-L. Chen, "Wafer map failure pattern recognition and similarity ranking for large-scale data sets," *IEEE Tran. on Semi. Manufacturing*, vol. 28, no. 1, pp. 1–12, 2015.

[2] Y. J. Zeng, L.-C. Wang, and C. J. Shan, "Miniature interactive offset networks (minions) for wafer map classification," in *IEEE International Test Conference*, 2021, pp. 190–199.

[3] N. Sumikawa, M. Nero, and L.-C. Wang, "Kernel based clustering for quality improvement and excursion detection," *IEEE International Test Conference*, 2017.

[4] M. Fan, Q. Wang, and B. van der Waal, "Wafer defect patterns recognition based on optics and multi-label classification," *IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2016.

[5] J. Yu and X. Lu, "Wafer map defect detection and recognition using joint local and nonlocal linear discriminant analysis," *IEEE Tran. on Semi. Manufacturing*, vol. 29, no. 1, pp. 33–43, 2016.

[6] a. a. Minghao Piao, "Decision tree ensemble-based wafer map failure pattern recognition based on radon transform-based features," *IEEE Tran. on Semi. Manufacturing*, vol. 31, no. 2, pp. 250–257, 2018.

[7] N. Yu, Q. Xu, and H. Wang, "Wafer defect pattern recognition and analysis based on convolutional neural network," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 4, pp. 566–573, 2019.

[8] J. Wang, Z. Yang, J. Zhang, Q. Zhang, and W.-T. K. Chien, "Adabalgan: An improved generative adversarial network with imbalanced learning for wafer defective pattern recognition," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 3, pp. 310–319, 2019.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and J. Bengio, "Generative adversarial networks," *arXiv:1406.2661*, 2014.

[10] T.-H. Tsai and Y.-C. Lee, "A light-weight neural network for wafer map classification based on data augmentation," *IEEE Transactions on Semiconductor Manufacturing*, vol. 33, no. 4, pp. 663–672, 2020.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pp. 318–362, 1986. [Online]. Available: https://doi.org/10.21105

[12] M. Saqlain, Q. Abbas, and J. Y. Lee, "A deep convolutional neural network for wafer defect identification on an imbalanced dataset in semiconductor manufacturing processes," *IEEE Transactions on Semiconductor Manufacturing*, vol. 33, no. 3, pp. 436–444, 2020.

[13] M. B. Alawieh, D. Boning, and D. Z. Pan, "Wafer map defect patterns classification using deep selective learning," *ACM/IEEE Design Automation Conference*, 2020.

[14] H. Hu, C. He, and P. Li, "Semi-supervised wafer map pattern recognition using domain-specific data augmentation and contrastive learning," in *2021 IEEE International Test Conference (ITC)*, 2021, pp. 113–122.

[15] Y. J. Zeng, L.-C. Wang, C. J. Shan, and N. Sumikawa, "Learning a wafer feature with one training sample," in *IEEE International Test Conference*, 2020, pp. 1–10.

[16] F.-F. Li, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[17] M. Fink, "Object classification from a single example utilizing class relevance metrics," *Advances in Neural Information Processing Systems*, pp. 449–456, 2005.

[18] L.-C. Wang and J. Zeng, "Machine learning support for wafer-level pattern analytics," *Chapter 9 in Machine Learning Support for Fault Diagnosis of System-on-Chip, Springer Nature*, 2023.

[19] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[20] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013. [Online]. Available: https://arxiv.org/abs/1312.6114

[21] S. Zhao, J. Song, and S. Ermon, "Infovae: Information maximizing variational autoencoders," 2017. [Online]. Available: https://arxiv.org/abs/1706.02262

[22] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[23] B. Schölkopf and et al., *Learning with Kernels:Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2001.

[24] M. Dupree, M. J. Yang, Y. J. Zeng, and L.-C. Wang, "Iea-plot: Conducting wafer-based data analytics through chat," in *IEEE International Test Conference*. IEEE, 2023.

[25] X. Dong and J. Shen, "Triplet loss in siamese network for object tracking," in *European Conference on Computer Vision*, Sep 2018.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.