DEEP LEARNING BASED VIRTUAL METROLOGY IN SEMICONDUCTOR MANUFACTURING PROCESSES

by

HARSHIT BOKADIA

A thesis submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Master of Science

Graduate Program in Industrial and Systems Engineering

Written under the direction of

Myong K. Jeong

And approved by

_____

_____

_____

_____

New Brunswick, New Jersey

OCTOBER, 2018

ABSTRACT OF THE THESIS

Deep Learning Based Virtual Metrology in Semiconductor Manufacturing Processes

By HARSHIT BOKADIA

Thesis Director:

Myong K. Jeong

Virtual metrology (VM) in semiconductor manufacturing is the technique of predicting critical dimensions of wafer quality characteristics without direct measurement based on process data of production equipment. VM is important in semiconductor manufacturing since it enables engineers to monitor the quality of wafers in production without physical wafer metrology thereby increasing the throughput of the process. As the process information consists of a large number of process variables in the form of raw sensor signals, learning new useful features in a low dimensional space is a key to build accurate VM prediction models. Earlier efforts in VM modeling were carried out by employing linear dimensionality reduction techniques such as PCA. Autoencoder is a deep learning based feature extraction method that has the capability to explore the non-linearity in the modeling and to represent high dimensional input into a low dimensional space. In this thesis, we propose a new VM model by incorporating the autoencoder based feature learning. We apply the proposed model to the prediction of critical dimensions of wafers at a plasma etching process in semiconductor manufacturing and compare the predictive

performance of the proposed model with conventional VM models. The experimental results show that the proposed model outperforms the existing models thus showing that autoencoder based feature learning is helpful in VM modeling with raw sensor signals.

Acknowledgement

I would like to take this opportunity to acknowledge everyone who had faith in this research work.

First and foremost, I would like to express my gratitude to my committee chairperson Professor Myong K. Jeong for his constant support, guidance and motivation in completing this thesis. He taught me the way to approach a research problem and contribute effectively in the research. He had been very patient and supportive throughout the research. He has the insight of an accomplished scholar and provided valuable feedbacks during his reviews which helped me in making substantial contribution to this research. I am grateful for his support.

The assistance and guidance of my fellow PhD student Jeongsub Choi had been very much essential in completing this thesis. He helped me in writing python scripts and implementing the ideas throughout this research work. He is a very hard working researcher and had been of tremendous help despite his busy schedule. I appreciate his support and guidance.

I would also like to thank the committee members Professor Weihong Guo and Professor Hoang Pham for taking out time from their busy schedule and providing valuable feedback on my thesis.

I would like to thank the faculty, staff, and students of my department, Industrial and Systems Engineering at Rutgers for their continuous support during my time at Rutgers.

At last, I would take this opportunity to thank my parents and sister for supporting me for this research work and keeping me motivated throughout the journey.

Table of Contents

List of Tables

List of Figures

CHAPTER 1 Introduction

The manufacturing of integrated circuits (ICs) also known as chips in semiconductor manufacturing industry involves a large number of sequential steps such as chemical vapor deposition, etching process and lithography process. During each step, patterned layers are being formed on the wafers which eventually complete the integrated circuit and we get the final product called ICs or the chip. From each wafer, hundreds of chips are being formed. Definitely, the complete process involves a huge amount of capital expenditure. To ensure cost effectiveness and process stability, an efficient and reliable method for process control is required and since semiconductor manufacturing consists of a number of sequential manufacturing stages, quality control at each stage is important for high production yield.

High yield in production and product's quality can be achieved by good quality management. A way for quality control is to monitor manufacturing process by measuring critical dimensions of wafer quality characteristics at various steps during the process (metrology). Another way is to promptly detect wafer abnormalities by identifying spatial defect patterns (Kim *et al*., 2016). Fail bit maps (FBMs) are also quite commonly used tools for monitoring wafer quality which involves visual inspection of failed cell count during wafer functional tests. Some researchers proposed an automated classification procedure for failure pattern on FBMs using regularized singular value decomposition which no more requires any visual inspection thus saving time and money (Kim *et al*., 2015). Both metrology and fault detection is important in achieving a better overall quality control. Traditionally, manufacturing processes in semiconductor industry

were controlled using statistical quality control techniques and metrology being one of them has been in practice for a long time (Su *et al*., 2007). Critical dimensions of wafer quality characteristics are being measured at various stages during the process by sampling few wafers out of the lot (Su *et al*., 2007; Khan *et al*., 2007). These measurements were done with physical metrology tools which produce a lot of delay in providing feedback of the current wafer quality (Chang and Cheng, 2005). Also, this physical metrology is being performed only on monitor wafers which are drawn intermittently during the the process (Kang *et al*., 2011). Not only this, the equipment condition keeps on changing due to malfunction, preventive maintenance events, and equipment aging problems. This changes the process parameters and hence the wafer quality. These process shifts and drifts in equipment condition which occur between the measurements are not easily detected which may lead to faulty wafers thereby decreasing the cost efficiency.

Virtual metrology (VM) in semiconductor manufacturing overcomes the limitations of conventional physical metrology by predicting critical dimensions of wafer quality characteristics without direct measurement based on process data of production equipment (Chang and Cheng, 2005). It enables engineers to monitor the quality of wafers in production without physical wafer metrology thereby increasing the throughput of the process. This is the reason VM is gaining increasing attention from semiconductor manufacturing industry.

There are several advantages of shifting to virtual metrology from physical metrology in semiconductor manufacturing. Earlier, for metrology operation, the quality statistics were collected for only a few wafers from the lot (Khan *et al*., 2007). It was

hard to measure every production wafer since it is time-consuming and expensive also. These few wafers were assumed to represent the product quality of all the production wafers (Hung and Lin, 2007). VM provides more information on the quality of processed wafers in production rather than just a few sampled wafers since for VM modeling few wafers are sampled from the lot to get metrology data (response variable) and the metrology values for the rest of the wafers are predicted through VM prediction models. VM enables to detect the shifts and drifts occurring due to changing process dynamics immediately and reconfigure the process parameters accordingly in real time thereby preventing the production of faulty wafers (Lin *et al*., 2006). VM drives the production efficiency and eventually throughput by enabling the process control and monitoring in real time (Khan *et al*., 2007; Qin *et al*., 2006). Also, with increased complexity of circuit design and chips getting smaller and smaller, it is required to shift to the wafer to wafer control from earlier lot to lot control which is possible by virtual metrology because of faster response and no metrology delays.

For VM, the process information is collected from the sensors mounted on the production equipment, and the resultant data can be very high dimensional. Existing VM modeling techniques have some limitations when we deal in the high dimensional space since they consider linear dimensionality reduction techniques such as PCA (Kang *et al*., 2011; Hung and Lin, 2007; Zeng and Spanos, 2009; Kurz *et al*., 2015; Park *et al*., 2015) and least angle regression (Susto and Beghi, 2013). But, the functional data coming from different sensors might have a non-linear mapping with the response variable and performing linear dimensionality reduction overlooks this non-linearity. On the other hand, some VM models are based on features from variable selection such as stepwise

selection, statistical filtering and various other variable selection techniques (Purwins *et al*., 2011; Zeng and Spanos, 2009) which eventually leads to losing of some important process information from the data due to reduction in model complexity. However, deploying existing VM techniques directly in the high dimensional space may lead to a poor generalization.

Autoencoder (Vincent *et al*., 2008) and its variants with deep neural network structures have this capability of learning non-linear features from the data in a low dimensional space (Hinton and R.Salakhutdinov, 2006). Deep neural networks have shown successful performance at many complex modeling tasks such as fault detection and classification in semiconductor manufacturing (Lee *et al*., 2017), fault diagnosis in rotating machinery elements (Jia *et al*., 2016), reciprocating air compressors (Tran *et al*., 2014) and at representation learning (Bengio *et al*., 2013).

So, the alternative may be to do dimensionality reduction taking into account the non-linear mapping of process variables with the metrology values. Our motivation for this research is in developing a new VM model which encompasses this feature learning with autoencoders to improve the VM performance. Deep learning models have the ability of mapping approximate non-linear function between predictor variables and response variable through multiple hidden layers and non-linear activation functions (Larochelle *et al*., 2009).

If we learn good features in a lower dimensional space by taking non-linearity into account through a deep learning based feature extraction method known as autoencoders, we have a good chance of getting higher prediction accuracy for various existing prediction models.

In this thesis, we proposed a new VM modeling technique using deep learning based feature extraction models. The objective in this thesis is to show that features extracted by autoencoder and its variants with deep structure improve the VM performance where autoencoders are capable of learning non-linear features from data as well as reducing the dimensionality. We evaluate the proposed methodologies by conducting two different experiments on a real-life dataset of a plasma etching process in semiconductor manufacturing. The first experiment is to evaluate the methodology with deep learning based feature extraction by applying autoencoders on conventional VM approach of process statistics data. The second experiment is to apply autoencoder directly to raw sensor data. In both the case, the features learned by autoencoder improves the VM performance.

In chapter 2, we review the literature of VM and autoencoder and briefly explain the key concepts on deep learning. Chapter 3 explains the new proposed VM model. In chapter 4, we evaluate the proposed model using a real-life sensor dataset. Chapter 5 concludes this thesis and states future research work.

CHAPTER 2 Related Work

2.1. VM in Semiconductor Manufacturing

VM is for the prediction of metrology values or critical dimensions of wafer quality characteristics from process information which is generally present in the form of functional data from sensors mounted on the production equipment. The purpose of VM is to provide metrology values of all the wafers present in the lot thus enabling wafer to wafer control rather than lot to lot control. The Fault detection and classification (FDC) data and metrology data which are used for process control can be used for VM modeling (Cheng and Cheng, 2005; Chang *et al*., 2006). VM system is in short part of the process control along with FDC system.

VM modeling belongs the advanced process control (APC) techniques rather than traditional statistical process control techniques due to real-time control feedback. Being a kind of APC techniques, VM modeling exploits the historical data in the form multivariate process information to predict metrology values with real-time configuration of process parameters in order to maintain product quality. VM modeling consists of data processing as the very initial step which involves data cleaning in the form of outlier detection, imputation of missing values and statistical analysis of data by dimensionality reduction in the form variable selection or feature transformation to a lower dimension. It also involves correlation analysis of process variables to remove redundant variables and keep only the significant ones. This stage of data pre-processing assures the quality of data we input to the VM prediction models by keeping only that much of process information which truly explains the variance with target variable i.e. metrology data in

case of VM. The next step is to test the performance of various VM prediction models and compare the model performance by a suitable performance measure to assess the best model suitable for VM modeling. Various approaches have been taken by the researchers in the past at all these different stages of VM modeling.

First, let's discuss the previous work by researchers at the data preprocessing stage for VM modeling. To begin with, a considerable amount of work has been done regarding outlier detection. Outlier detection is an important step in VM modeling. There is a possibility of the faulty wafer being produced due to process shifts and drifts happening. This faulty wafer may cause the metrology values to have an extremely high or low value thereby affecting the overall predictive performance of the model. Thus, it is important to have these outliers detected and removed. Zeng and Spanos (2009) deal outlier detection in VM modeling where they treated the points with distance metric value greater than the cut-off value as outliers and adopted Mahalanobis distance as the distance metric. Some researchers adopted Hotelling T^2 test in their work on VM modeling for dealing with outliers (Hirai and Kano, 2015; Zeng and Spanos, 2009). Another paper proposed a robust regression model based on relevance vector machine to deal with outliers by using weight strategy (Hwang *et al*., 2014). Though not specific to VM, recently Hwang *et al*. (2015) proposed a robust kernel-based regression technique which along with estimating the non-linear functional relationship between predictor and response variable can also help in dealing with outliers in both X and Y space which may prove to be helpful in VM applications.

Variable selection is a major challenge in VM modeling due to a high number of process variables involved. Finding critical or important process variables which explain

the variations in output is significant in building accurate prediction models. Various methods were proposed by researchers in the past for variable selection. Hung and Lin (2007) adopted stepwise selection to extract critical parameters. They also adopted Principal Component Analysis (PCA) and Bartlett test for further reducing input variables to their prediction models. Though not specific to VM, recently a novel PCA based neural network approach adopting Bayesian information criteria for studying high dimensional multivariate functional data was proposed to compare the prediction models efficiency in semiconductor manufacturing (Ko *et al*., 2013). Pampuri *et al*. (2011) proposed multilevel lasso algorithm and used L1 penalization to obtain models of lower order. Purwins *et al*. (2011) used only simple statistical filtering and expert selection for variable selection. Zeng and Spanos (2009) adopted random modeling genetic partial least square algorithm and stepwise selection for variable selection. Hirai and Kano (2015) adopted variable importance in projection (VIP) method for doing the variable selection. Susto and Beghi (2013) proposed another technique called least angle regression for model selection.

Variable selection techniques are mainly concerned with dimensionality reduction. At the same time, we also need to retain information from important variables which may be dropped while employing some variable selection technique. This is where regularization comes into the picture since it helps in penalizing the coefficients rather than completely getting rid of the variable. The basic approach in a regression technique is towards minimizing the residual sum of squares which may sometime lead to overfitting. This led to the development of regularized models such as that of ridge regression. Still, the problem of high dimensional data is not solved since the ridge

regression does not perform the dimensionality reduction. As a result, L1 penalized techniques were developed which helps to obtain the sparse solution in case of the smaller value of regularization parameter. A study of Pampuri *et al*. (2011) proposed a multilevel lasso technique which is basically an extension of L1 penalization to a hierarchical framework. Recently a technique which provides a sparse explicit description of the function in input space along with exploring non-linearity in the data was proposed (Lee *et al*., 2014). The technique can be effective in VM application since it takes care of the non-linearity in modeling, tackles the variable selection issue in kernel-based approaches to some extent and in addition is less sensitive to a range of data.

The next stage after data pre-processing involves working with prediction models for VM. Many machine learning based models were proposed by researchers for VM modeling such as support vector machine for regression (Purwins *et al*., 2011; Chou *et al*., 2010; Kim *et al*., 2017), relevance vector machine with variational inference for regression (Hwang *et al*., 2014) and K- nearest neighbor for regression (Lee *et al*., 2014). Some researchers used linear modeling techniques such as multiple linear regression and lasso regression in VM modeling for a single output variable (Susto and Beghi, 2013), others proposed VM models based on partial least squares regression for multiple output variable (Hirai and Kano, 2015; Khan *et al*., 2007; Purwins *et al*., 2011). Another approach for VM modeling currently is that of neural networks and some researchers proposed feed-forward network with error Backpropagation for VM modeling (Su *et al*., 2006; Zeng and Spanos, 2009; Cheng and Cheng, 2005). Some others proposed radial basis function neural network (Ferreira *et al*., 2009; Hung and Lin, 2007).

2.2. Autoencoder and Deep Neural Networks

Artificial neural networks are computational models as mimics of biological neural networks in a human brain. They consist of neurons interconnected with each other in multiple layered networks. These networks are capable of non-linear modeling with the help of non-linear activation functions at the neurons. The neurons on an input layer take input data, and the neurons on the subsequent layers take the output of the activation function from the preceding layers. A deep neural network consists of a multiple numbers of hidden layers in the network and is trained in a hierarchical fashion. i.e. features learned by subsequent layers depend on those of the previous layers. There are various kinds of different deep neural network models in use today and together they comprise the field of deep learning.

A simple artificial neural network consists of one or two hidden layers in general. The paper on universal approximation theorem proved that even shallow neural networks with as small as one hidden layer having sufficient neurons can approximate any complex function (Hornik *et al*., 1989; Cybenko, 1989). It is possible that the functions that are represented well by using deep nets may require almost an exponential number of neurons with a shallow network. But still, we required deep learning models because sometimes single layer or shallow network is not able to solve various problems like object recognition, image recognition, speech recognition etc. since the learning process is hierarchical i.e. each layer learns certain features and then the subsequent layers build on the output of previous layers.

Autoencoders belong to a family of deep learning models where output layer is exactly the same as the input layer. Autoencoders aims to learn representation for a set of

input data typically in low dimensional space (Hinton and R.Salakhutdinov, 2006). It does so by first compressing the input to a low dimensional space through 'encoder' and then reconstructing the input through 'decoder'. The low dimensional representation of the input (encoder) is also called 'latent space representation'. This network is typically trained through a well-known algorithm called Backpropagation for optimizing the cost function to obtain parameters i.e. minimize the reconstruction error or loss function. Autoencoder is basically an unsupervised deep learning algorithm since it does not require any target values or labels to train the network. It is, in fact, a kind of self-supervised algorithm as it takes the raw input and generates the labels by forming the output layer exactly same as the input layer. Figure 2.1 below shows a basic one hidden layer autoencoder.

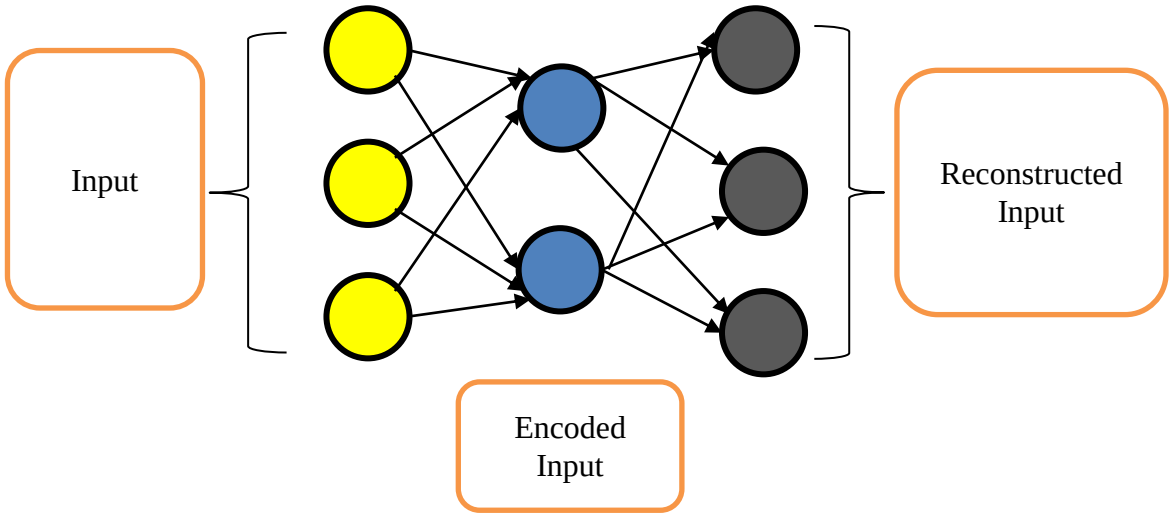

Figure 2.1 Basic one hidden layer autoencoder

Different strategies of deep learning models are applicable in case of autoencoders also like training by Backpropagation, introducing sparsity or regularization constraint, avoiding overfitting by dropout (Srivastava *et al.*, 2014), using different

optimization algorithms for faster convergence. We can stack multiple basic autoencoders to create a deep autoencoder to learn more complex functions. We are going to review few important concepts of deep neural networks in the further sections which are very much applicable to autoencoders. By using these techniques, we can create various different autoencoders. For example, adding regularization constraint to the layer in the form of L1 or L2 penalization may help in avoiding overfitting while training the autoencoders. This regularization term forces the autoencoder to learn the input representation through a small number of nodes by penalizing some nodes to zero. This sometimes helps in learning better features. Using convolutional layer instead of dense hidden layer creates a convolutional autoencoder. Similarly stacking multiple autoencoders creates a deep autoencoder and many others.

Autoencoder are different from PCA or other dimensionality reduction techniques in the manner that autoencoders can induce non-linearity in the networks through its activation functions i.e. we can have a non-linear transformation of data while PCA is all about linear transformation

An important concept in solving any machine learning problem is that of bias and variance. In simple terms, bias refers to the training error and variance to the test error. We generally want to reduce the variance without affecting bias. But, in practicality, there is always some trade-off between the two which is known as the bias-variance trade-off. Recently deep learning pioneers in their book (Goodfellow *et al.*, 2016) discussed the concepts of regularization and overfitting in detail. A very high bias refers to ‘underfitting’. This means the model has converged to a suboptimal solution and has not learned the function mapping properly or rather has not explored the possible relationship

between the predictor and response variable effectively. The solution can be to try a deeper architecture or a deep autoencoder in our case to learn new features, in short, increasing the model capacity. Now, as we increase the model capacity by deploying a deeper architecture, we may reduce our training error to a very low value but it may lead to a bad generalization error. This is the case of high variance or 'overfitting'. In this case, the model is mapping the relation between predictor variable and response variable very accurately for training data but does not generalize well enough on test data. To overcome this issue of overfitting we generally introduce various regularization techniques in our autoencoder models while learning new features such as L1 and L2 sparse autoencoders as mentioned before. Regularization helps in decreasing the variance though there may be some trade-off with bias.

Regularization in deep learning or any machine learning technique is a way of improving model performance by reducing the generalization error but not the training error. During overfitting, the model learns too much of details or we can say it overlearns with the noise in the data and as a result, it does not generalize well enough on a new data or unseen data. Some of the commonly used techniques for regularization in autoencoders are L1 or L2 regularization. We can create new autoencoders by adding L1 & L2 regularization constraint in a basic autoencoder to learn better features for our prediction models.

Optimization also plays an important role in deep learning models since they try to minimize the loss function (regression model as in our case) and helps in finding the best model parameters. The optimizer we choose for our autoencoder determines the quality of features we have learned since the way the model parameter values are updated and

optimized during a training process depends a lot on various optimization algorithms used in deep learning. A recent paper on gradient descent optimization in deep learning gave some insights into many optimization techniques in the deep learning area (Ruder, 2016). Majority of the optimization algorithms in deep learning are based on first order optimization since computing a second order derivative is very costly. Though second order optimization also tells about the curvature of loss function and it shows if the first order derivative is increasing or decreasing. Therefore, second order derivative does not get stuck in local minima since it knows about the curvature of the loss function.

The equation for parameter update in a typical neural network is as:

$$w = w - L_R . \nabla(w) E(w) \tag{1}$$

where: L_R stands for learning rate, $\nabla(w)E(w)$ for gradient of loss function $E(w)$ with respect to w and w is the weight parameter.

From equation (1), we can see that the weights are updated in the opposite direction of the gradient. The first order minimization often leads to a local minimum. Generally, more complex problems have a non-convex function mapping. Though a deep neural network may approximate this highly non-convex function mapping but it may converge to local minima. The parameter updates were done only once for a batch gradient descent method. This was however not considered good since it had only one update and problem of getting stuck in a local minimum was high and also it had a slow convergence. This was rectified by stochastic gradient descent (SGD) method and equation (1) now become:

$$w = w - L_R . \nabla(w) E(w; i) \tag{2}$$

where i is the training observations which is stochastically chosen. Now, SGD has definitely more parameter updates as compared to a traditional batch gradient descent algorithm. This may cause high variance and may lead to overshooting the minima.

Now, a proper selection of learning rate also affects the training of our autoencoder to a large extent. In fact, the way parameters are updated based on learning rate is the key difference between various optimization algorithms. The paper by Ruder (2016) discussed various adaptive algorithms. The 'Adagrad' algorithm allows the learning rate to adapt itself based on parameter update. It no longer has a fixed learning rate as SGD. However, it suffers from a problem of decaying learning rate which ultimately leads to a very slow convergence. In 'Adadelta' optimization algorithm this problem is decaying learning rate is mitigated. Another algorithm 'Adam' stands for adaptive moment estimation. 'Adam' actually helps in adapting learning rate and momentum according to the parameter updates. We calculate the first moment and second moment which is basically the mean and variance respectively and based on that we update the parameters. Detailed 'Adam' algorithm can be referred from the original paper (Kingma and Ba, 2015). However, there is no general rule to which optimization algorithm may work best and we may need to try different optimization methods to get the best result.

In autoencoder or any kind of deep neural network model, the parameters are basically the weights and biases which are learned during the model training process. Other than this, there are some parameters which are not learned during the training process and these are called as hyper-parameters. These hyper-parameters actually control the learning of the parameters which are learned during the training i.e. weights and

biases. Some of the important hyper-parameters are learning rate, momentum, choice of activation function, number hidden layers and number of units in each hidden layer, mini-batch size, optimizers, regularization parameters, decay rate etc. to name a few. Hyper-parameters are the parameters which are not learned from the data during the training process. In fact, they are external to the model training and are often chosen outside of the learning process. These hyper-parameters need to be optimized in order to get optimum values for model parameters. We often tune in the hyper-parameters in a way so as to get the best possible values for the model parameters. There are various ways to tune in those hyper-parameters viz. grid search, random search and Bayesian optimization. In grid search, we select some points on the hyper-parameter range which are uniformly distributed across the whole range. However, in this type of search, we train the neural network using all possible combinations of hyper-parameters across the grid and we might be training our deep net on a range of hyper-parameters which are not suitable. Instead, if we rather do the random search, we select the values for hyper-parameters randomly across the grid rather than using the uniform scale and then we narrow down our search for the hyper-parameters values which works well enough (Bergstra and Bengio, 2012).

CHAPTER 3 Proposed Methodologies With Features Extracted by Autoencoder.

Our methodology is based on the essence of accurately modeling the VM prediction problem for a very high dimensional dataset by retaining most of the information. So, to reduce the dimension taking non-linearity into account, we adopt a deep neural network based feature extraction method known as autoencoder which learns new features in a low dimensional space by reconstructing the input.

We propose two methodologies for VM incorporating feature learning through autoencoder. Firstly, we adopt autoencoder for feature extraction from process statistic data, on which conventional VM modeling approaches have been based, and the extracted features are used for the prediction in VM. To convert the raw sensor dataset into process statistics dataset, various statistics like mean, median, skewness, kurtosis etc. to name a few are being used. Then we learn new features on this process statistics data with the help of different autoencoders and use the extracted features for prediction modeling. On the other hand, we propose the other approach that autoencoder is employed for feature extraction directly from raw sensory data, and the extracted features are used for the prediction in VM. Both the methodologies are being discussed in detail in section 3.1 and 3.2 respectively.

In both the methodologies, we fix the encoding dimension for our autoencoder to learn new features. For different encoding dimensions in a one hidden layer autoencoder, we calculate the reconstruction error by computing mean squared error (MSE). The MSE value will slow down in improvement after a certain value of hidden neurons. We adopt that number of hidden neurons in the encoder layer as our input dimension for the

prediction models. This method helps in reducing the dimension and at the same time also helps to retain a reasonable amount of information for the functional sensor data by mapping non-linear functions.

3.1 VM with Features from Process Statistics.

Typically we are given a raw sensor data from a semiconductor fabrication process which is monitored by certain numbers sensors on the process equipment, and which consists of number of recipe steps also called sub-operations. The signals from the process equipment sensors are stored over the time duration and so the sensors provide the signals of T time points for each wafer. A critical dimension of wafer quality characteristics is measured after being processed.

In the first methodology, we convert the raw sensor dataset into a design matrix by extracting certain useful features such as summary statistics like mean, variance, standard deviation, skewness, kurtosis etc. There are a few advantages of extracting features from these signals. First, features extracted from signals are more robust to noise as compared to signal itself. Secondly, sometimes the precise value of the signal at a point of time may not make as much sense as the trend in the signal over the time. Third, the process information may be stored in a smaller memory space making the statistical analysis easier and faster. Let us suppose we extract Q statistics from each of the R sensors for each of the S sub operations. Thus, now our new process statistics dataset has $Q * R * S$ number of predictor variables which is equal to 'p' i.e. total number of predictor variables as stated earlier in the section. Conventionally, this process statistics dataset which is high dimensional is being used for prediction with VM.

In our methodology, we learn new features on this process statistics dataset using autoencoder and its variants in a low dimensional space with non-linear activations at the hidden units. Then, we use these new features for deploying VM prediction models. This methodology is summarized in the Figure 3.1 below.

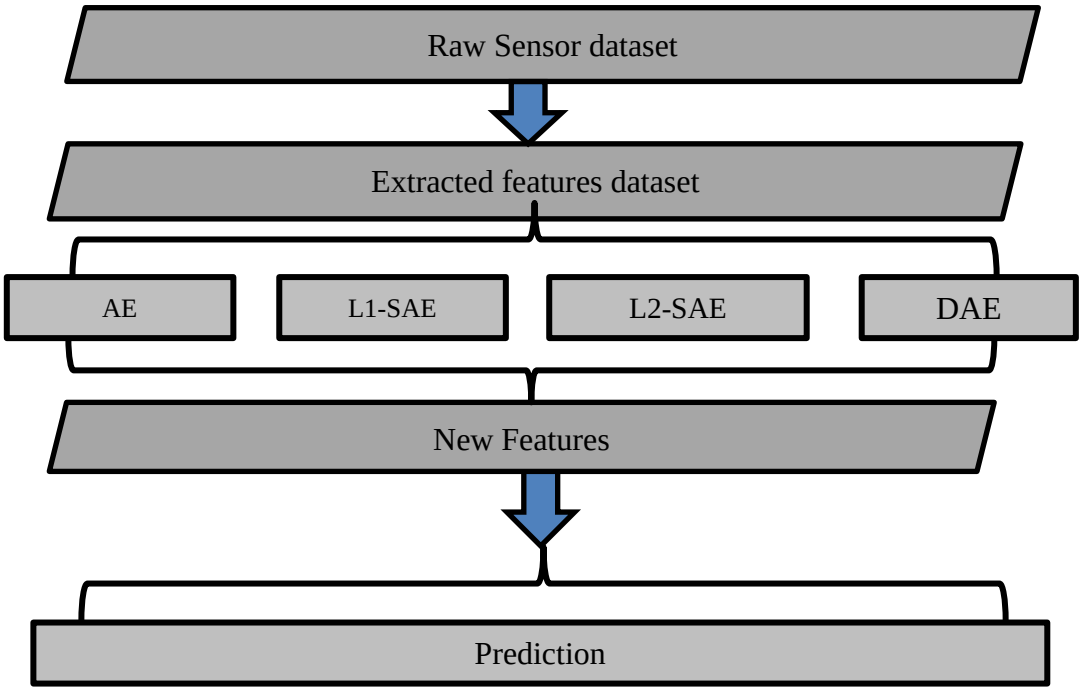

Figure 3.1 Methodology for VM based on feature learning by autoencoder on process statistics data.

Suppose that, for each wafer, the fabrication process is monitored by R sensors on the process equipment over time T. Consider there are S sub-operations also called recipe steps for the process. For sub-operations, $s = 1,2, \dots S$; the signals from the sensors are stored over the time duration T_s, and so the sensors provide the signals of T time points for

each wafer where $T = \sum_{s=1}^{S} T_s$. Then Q summary statistics such as mean, standard deviation, skewness, kurtosis etc. are extracted from the signal of each sub-process.

Let $\{\mathbf{z}_i, t_i\}_{i=1}^{n}$ be a dataset from N observed wafers where $\mathbf{z_i}$ are p-dimensional input vectors and t_i are output values for i = 1, 2, …, n. To be specific, each $\mathbf{z}_i$ consists of Q summary statistics from raw signals for the i-th wafer concatenated in a vector $\mathbf{x}_i$ for R sensors as $p = QRS$, and t_i is a measured critical dimension from a fabrication process. In our case of semiconductor manufacturing, the number of observations n are basically the number of wafers.

An autoencoder is trained through Backpropagation for reconstruction of inputs i.e. the output layer, in this case, is same as the input layer. Let us consider a general feed-forward neural network with L hidden layers and N_L neurons in the L^{th} layer. The output vectors at the m^{th} layer $\mathbf{a}^{(m)} \in \mathbb{R}^{N_m}$ for m = 0, …, L are computed as:

$$\mathbf{y}^{(\mathrm{m+1})} = \mathbf{W}^{(\mathrm{m+1})}\mathbf{a}^{(\mathrm{m})} + \mathbf{b}^{(\mathrm{m+1})} \tag{3}$$

$$\mathbf{a}^{(\mathrm{m+1})} = f_{(m)}\left(\mathbf{y}^{(\mathrm{m+1})}\right) \tag{4}$$

where, $\mathbf{W}^{(m)} \in \mathbb{R}^{N_m \times N_{m-1}}$ is a weight matrix, $\mathbf{b}^{(m)} \in \mathbb{R}^{N_m}$ is a bias vector, and $f_{(m)}$ is an element-wise activation function on the m^{th} layer. $\mathbf{a}^{(0)}$ indicates the given input vector $\mathbf{z} \in \mathbb{R}^{N_0}$ and the output vector at the L^{th} hidden layer is $\mathbf{a}^{(L+1)} = f(\mathbf{z}; \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the set of all the weight parameters in f. Now we need to learn a new representation of p-dimensional input $\mathbf{z}$ into a lower dimension space through autoencoders. That is, the input vector $\mathbf{z}$ is taken in the autoencoder as $\mathbf{a}^{(0)}$ in (3). In case of a single-hidden-layer

autoencoder, we get the reconstructed input vector $\mathbf{a}^{(2)}$ at the output layer given by (3) and (4). Particularly, $\mathbf{a}^{(1)}$ becomes the extract feature vector at the hidden layer as the latent space representation based on the encoded input. To obtain the features, we train the autoencoder by minimizing the cost function $E(\boldsymbol{\theta})$ given by

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left\| \mathbf{a}_i^{(0)} - \mathbf{a}_i^{(L+1)} \right\|_2^2 \tag{5}$$

where $\mathbf{a}_i^{(m)}$ is the activation for the i-th observation on the m-th layer, and $\|\cdot\|_2$ is a L2 vector norm. So, we need to calculate min $E(\boldsymbol{\theta})$ i.e. find the values of parameters (weights) that minimize this cost function. We try to find these optimal values of parameters through an iterative algorithm known as the Backpropagation algorithm (Rumelhart *et al*., 1986). The Backpropagation algorithm updates the parameters by calculating the cost function at the last layer and backpropagating the error using chain rule through gradients of weights in a deep neural network.

We can also use other types of autoencoders to learn new features which we discussed in chapter 2. In addition to basic and deep autoencoder, we have the regularized autoencoder. In case of regularized autoencoder, the cost function $E(\boldsymbol{\theta})$ gets modified only keeping rest of the model as same. Now we can add a regularization constraint in the form of L1 and L2 penalization on the activation at the hidden layer of this autoencoder to form regularized autoencoder. This regularization term forces the autoencoder to learn the input representation through a small number of nodes by penalizing some nodes to zero.

Firstly, we add the regularization constraints on the activation at the hidden units of the autoencoder to learn new features with sparsity. For L1 sparse autoencoder this $E(\boldsymbol{\theta})$ gets modified due to the addition of regularization constraint and becomes:

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left\| \mathbf{a}_i^{(0)} - \mathbf{a}_i^{(L+1)} \right\|_2^2 + \lambda \sum_{i=1}^{n} \left\| \mathbf{a}^{(L)} \right\|_1 \tag{6}$$

where $\|\cdot\|_1$ is a L1 vector norm.This regularization term with L1 norm in (6) penalizes the activations at hidden units thereby producing a more compact representation. This compact representation can be more effective in getting better prediction performance than the basic autoencoder.

Similarly, for L2 norm autoencoder, the equation for loss function becomes:

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left\| \mathbf{a}_i^{(0)} - \mathbf{a}_i^{(L+1)} \right\|_2^2 + \lambda \sum_{i=1}^{n} \left\| \mathbf{a}^{(L)} \right\|_2^2 . \tag{7}$$

This regularized autoencoder also provides better features for prediction modeling by penalizing the activations at hidden units by a rather different norm. The L2 norm does not penalize the activations to an extreme zero value. This could be effective when sensor signals are highly correlated, since losing features due to extreme penalization as in the previous case can lead to the loss of important process information.

So, depending on the amount of correlation between the sensor signals, this kind of regularized variants of autoencoder can provide useful new features for prediction.

Once we have learned the new features, we use these features for predicting our metrology values with the help of various prediction models. The prediction models take

the data in the form output vector at the L^{th} hidden layer of our autoencoder which are basically our features learnt by autoencoder i.e. $\boldsymbol{a}^{(L)}{}_i$ for the i^{th} observation which is $\mathbf{v}_i$ for i = 1, 2, …, n. Let a prediction model be g and the output from the prediction model be $\hat{t}_i$ as

$$\hat{t}_i = g(\mathbf{v}_i). \tag{8}$$

The performance of various prediction models is then evaluated in terms of mean squared errors (MSE) through equation 9 where $\hat{t}_i$ being predicted output and t_i being the actual output.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{t}_i - t_i)^2 \tag{9}$$

3.2 VM with Features from Raw Sensor Dataset.

In our second methodology, autoencoder is employed for feature learning directly from the raw sensor data. In this, we first rearrange the dataset from raw sensor signals into a matrix for our regression problem by considering every time stamp for each sensor as a different feature. So, let's say there are R sensors and t_1 time stamps for each sensor (time stamps may be different for different sensors), so our rearranged raw sensor dataset has $p = Rt_1$ features or predictor variables.

In our previous methodology, we extracted statistics from raw sensor dataset which can lead to losing some important process information. In this methodology, we learn new features directly on the raw sensor dataset. Learning new features directly on

the raw sensor dataset keeps the process information intact and may provide better features than our first methodology.

We learn new features $\mathbf{v}_i$ from the p-dimensional raw sensor input vector $\mathbf{x}_i$ directly in a lower dimension space with the help of autoencoder. We learn $\mathbf{v}_i$ for i = 1, 2, …, n by getting output at the hidden layer through (3) and (4) by training our autoencoder through the Backpropagation. We can add the sparsity to the autoencoder while extracting features through (6) and (7) same as we did in our first methodology. The prediction models take the compact representation or the new features $\mathbf{v}_i$ for i = 1, 2, …, n as the input to get predicted output i.e. $\hat{t}_i$ with the help of (8) and we compare our model performances. This methodology summarized in Figure 3.2.

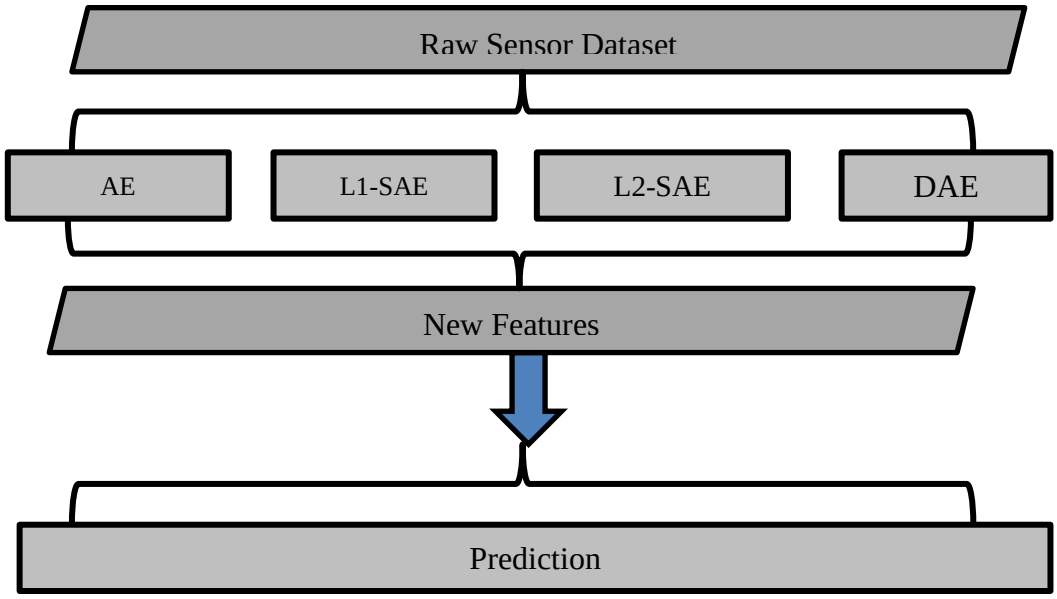

Figure 3.2 Methodology for VM based on features learning by autoencoder on raw sensor data.

CHAPTER 4 Experiments and Results

We consider an etching process in semiconductor manufacturing for our experimental work. Plasma Etching process is basically the removal of material from the surface by the plasma where the ions are bombarded on the surface of the wafer. It proposes various challenges like high dimensionality of data, equipment aging, changing process dynamics owing to maintenance activities, high correlation among various sensors, the introduction of outliers due to changing equipment conditions and various other challenges. The process also needs to be controlled in order to prevent etching of the following layers. So it is being monitored by various sensors measuring different physical parameters such as pressure, temperature, and others. Critical dimension values for various quality characteristics of the wafers e.g. “etch bias” in our case are being measured after completion of the etching process.

The proposed autoencoder based VM model in this thesis is evaluated on the basis of various experiments conducted on sensor dataset which is basically the data collected from a plasma etching process of a semiconductor manufacturing plant. The raw data from the sensor was collected over the same time length for a wafer i.e. the time length of all sensors for a particular wafer is same. However, sub-operations timing is different for different wafers. As a result, the total lengths of sensor signals are different for different wafers. The average length of the signals for the wafers in the dataset is 650 and the standard deviation of 10.22. The dataset consists of data from 85 sensors (these 85 sensors are picked out of approximately 200 sensors based on expert selection) for 298 wafers. The etching process equipment has 58 sub-operations. Figure 4.1 below shows a snap of the raw sensor data.

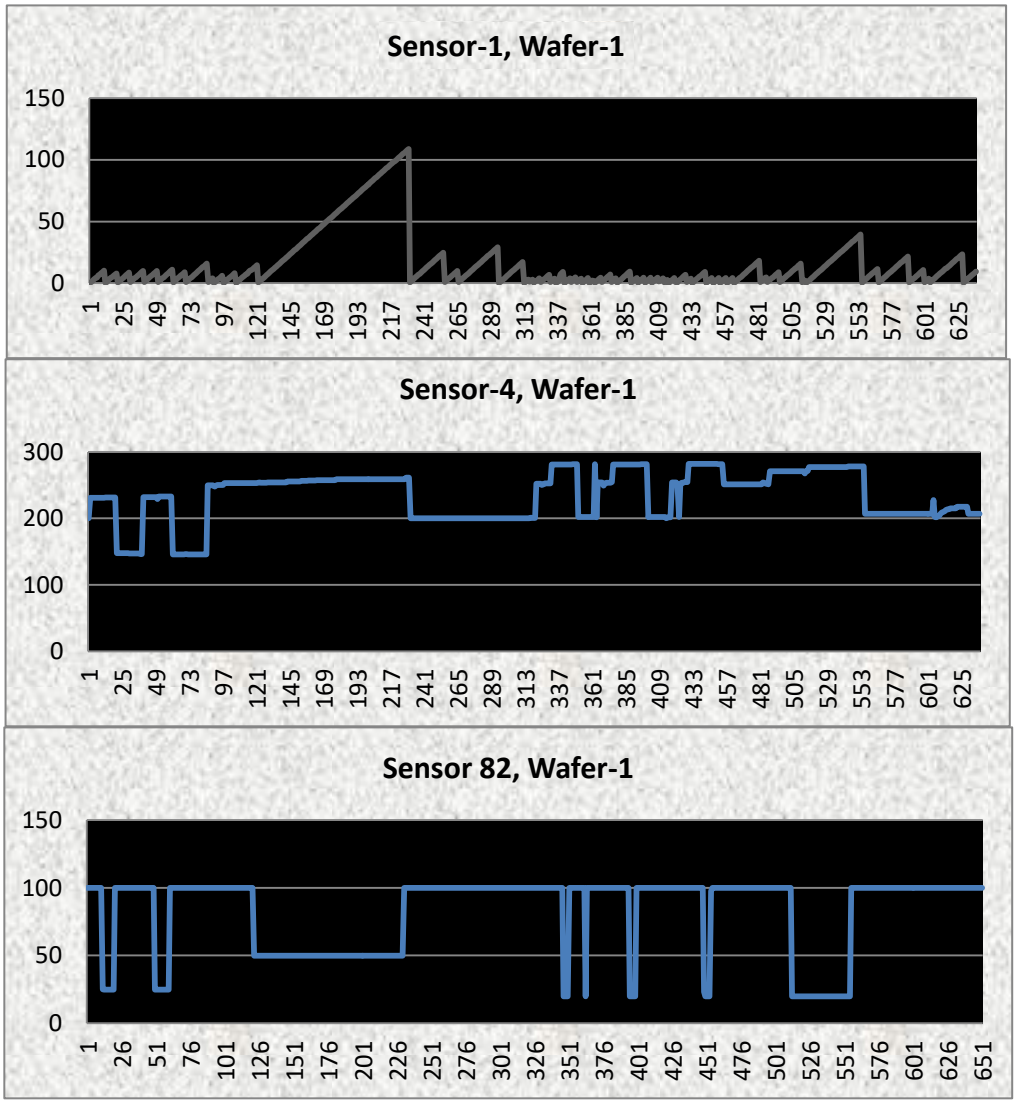

Figure 4.1 Raw sensor dataset from plasma etching process

We conducted two different experiments which will be explained in section 4.1 and 4.2 ahead. In our first experiment, we learned new features by four different autoencoders on process statistics data. In our second experiment, we learned new features directly on the raw sensor dataset. Then we deployed four different regression models on the new extracted features dataset in both the experiments for getting prediction performance. In both the experiments, we first ran an initial experiment with

varying number of neurons in the encoder layer for one hidden layered autoencoder to fix the encoding dimension for minimum reconstruction error. The rate of improvement of MSE decreases after a certain number of neurons so we fixed this as the encoding dimension for our autoencoders in both the experiments respectively.

The 298 observations are divided into two parts: 238 observations are used for model training and 60 observations are used for testing the model. 3- fold cross validation is adopted for the training set to avoid over-fitting and get a good generalization capability for the prediction model. Various set of hyper-parameters are tuned in by using grid search on this training set and the best combination of hyper-parameters are then tested on the test set to see if the model generalizes well enough.

We deployed four different regression methods i.e. SVM for regression, Ridge regression, Lasso regression and deep neural networks to evaluate the performance of our autoencoder based proposed method with existing methods. Support vector machine (SVM) for regression (Vapnik, 1999) is known for its good generalization capability. The objective is to minimize the sum of the ε-insensitive loss function. SVM needs the parameters 'C' for the regularization cost and '$\boldsymbol{\varepsilon}$' for error from the margins to be tuned in along with different kernel functions. We tested both linear and radial basis kernel to get the best solution with again 3-fold cross-validation. Lasso is another commonly used regression method. Lasso regression has also the capability of getting a sparse solution by penalizing some coefficient values to zero. The variable selection and regularization capability of Lasso by L1 penalization make it suitable to generalize well enough for regression task. The regularization parameter here is 'alpha' which is the only hyper-parameter fined tuned again with a 3-fold cross-validation. Ridge regression is found be

suitable for regression tasks which involve highly correlated variables. It invokes penalization by L2 norm. For deep neural network, a predetermined structure based on preliminary experiments done for fine tuning of different hyperparameters was adopted. The hyper-parameters to be tuned in for all the models are being summarized in Table 4.1 below.

Table 4.1 Hyperparameters to be tuned for various prediction models

Virtual Metrology Models	Hyper-Parameters to be tuned (3- fold cross validation used)
Support Vector Machine	C for regularization and type of kernel and its parameters
Lasso and Ridge Regression	α for regularization
Deep Neural Network	Learning rate, number of epochs, batch size, optimizer, dropout rate, number of hidden layers, number of neurons in each hidden layer.

To summarize, given the raw sensor data, we convert the raw sensor dataset into the design matrix for regression problem with respect to two different methodologies in our research. We then carry out the data preprocessing such as feature rescaling. A new representation of input features is learned by autoencoders for minimum reconstruction error. On this new transformed data, we deploy various prediction models such as SVM, Lasso, Ridge, deep neural networks to test the model performance. We do the fine-tuning of hyper-parameters for various prediction models with a randomized grid search and K-fold cross-validation to get the best configuration of the model with best parameters.

The computation for the experiments was implemented based on the python libraries: keras and scikit-learn.

4.1 VM with Autoencoder for Features from Process Statistics Data.

Nine different statistics shown in Table 4.2 were extracted from each sub-operation of each sensor signal for model implementation. Features are extracted from a discretized signal $\mathbf{z}$ of n observed time points. We have 85 sensors and 9 input statistics are extracted from 58 sub-operations of each sensor signal for the etching process. So, all together we have 85*9*58= 44370 input features and one output which is basically equal to the number of response variables (critical dimensions) which in our case is only one.

Table 4.2 Summary statistics extracted for each sensor signal

Feature	Expression
Length	n
Minimum	$\min(\mathbf{z})$
Maximum	$\max(\mathbf{z})$
Range	$\max(\mathbf{z}) - \min(\mathbf{z})$
Mean	$\frac{1}{N}\sum_i z_i$
Median	$\text{median}(\mathbf{z})$
Variance	$\frac{1}{n}\sum_i (z_i - \bar{z})^2$
Skewness	$\sum_i \left(\frac{z_i - \bar{z}}{\sigma}\right)^3$
Kurtosis	$\sum_i \left(\frac{z_i - \bar{z}}{\sigma}\right)^4$

On this process statistics dataset, we ran our first experiment for fixing the encoding dimension for our autoencoder as explained earlier. We get the following graph shown below and we fixed the encoding dimension as 2000 for our autoencoder as we can see

from Figure 4.2 that there is no substantial improvement in MSE after we increase our dimension from 2000 neurons.

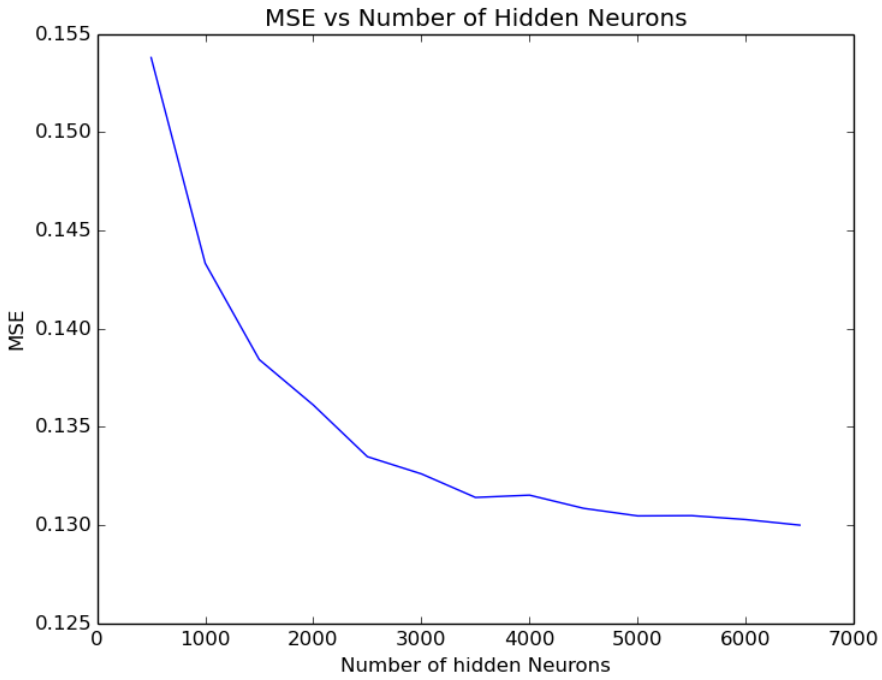

Figure 4.2 Graph of MSE and number of hidden neurons for process statistics dataset.

The Table 4.3 below show the testing MSE for various prediction models on the features learned by various autoencoders and we can see that the features learned by one hidden layered autoencoder provide better performance for various prediction models. We deploy prediction models on process statistics dataset (FE), on features learnt by one hidden layer autoencoder from process statistics dataset (AE+FE), on features learnt by L1 autoencoder from process statistics dataset (L1-AE+FE), on features learnt by L2 autoencoder from process statistics dataset (L2-AE+FE) and on features learnt by deep autoencoder from process statistics dataset (DAE+FE).

We also calculated the Pearson and Spearman correlation coefficients between features and the output variable for the new features learned by all different Autoencoders (AE) shown in Table 4.4 and Table 4.5 respectively. This gives an idea of the features learned by various autoencoders. The improvement in VM performance due to the new features learned by autoencoders can be attributed to correlation coefficients in the tables. In the following tables, *Pr_Mean* stands for mean of top 30 Pearson correlation coefficients. Similarly, *Pr_Max* stands for maximum of all the Pearson correlation coefficients. Likewise, *Sr_Mean* stands for mean of top 30 Spearman correlation coefficients and *Sr_Max* stands for maximum of all the Spearman correlation coefficients. Pearson correlation provides the measure of linear correlation between two variables. Table 4.4 shows that the features learned by one hidden layer autoencoder have reasonably comparable values of Pearson coefficients with process statistics dataset. But, it should be noted here that the autoencoder features are in a very low dimensional space with just 2000 features compared to almost 44000 features from process statistics dataset. In case of process statistics dataset, we can see that only 69 features are useful out of 44370 features which unnecessarily increase the model complexity thereby lowering the testing MSE. Also, if we calculate the ratio of critical features (i.e. features having correlation value >0.3 out of the total number of features), it is almost equal i.e. 0.0015. But as mentioned earlier we have a reduced model complexity in case of features learned by autoencoders by having more useful features in a low dimensional space. Hence, we can see the prediction models accuracy is better with features learned by autoencoder. Like Pearson correlation, Spearman correlation also measures the relationship between two variables. The difference between the two is that Pearson correlation coefficient

measures linear relationship between variables while Spearman correlation coefficient measures the monotonic relationship between variables. So, here we again see from the Table 4.5 below that the correlation coefficients improved for the features learned by autoencoder when we took non-linearity into account. This again explains why we have better prediction performance with the features learnt from autoencoder.

Table 4.3 Testing MSE for various prediction models

Models	LASSO	RIDGE	SVM	DNN
FE	0.150969	0.146868	0.138563	0.171452
AE+FE	0.147364	0.157370	0.127266	0.150757
L1-AE + FE	0.153619	0.161069	0.132117	0.167345
L2-AE + FE	0.153374	0.170123	0.136617	0.179518
DAE + FE	0.154102	0.182613	0.136749	0.196976

Table 4.4 Statistics for top 30 Pearson correlation coefficients

Models	Pr_Mean	Pr_Max	Number of coefficients> 0.3
FE	0.4361	0.4812	69
AE+FE	0.1990	0.4879	3
L1-AE + FE	0.2074	0.2898	0
L2-AE + FE	0.1917	0.3241	1
DAE + FE	0.1844	0.2399	0

Table 4.5 Statistics for top 30 Spearman correlation coefficients

Models	Sr_Mean	Sr_Max	Number of coefficients> 0.3
FE	0.3190	0.3573	55
AE+FE	0.2410	0.3178	1
L1-AE + FE	0.2385	0.2934	0
L2-AE + FE	0.2375	0.3180	1
DAE + FE	0.2386	0.3046	1

4.2 VM with Autoencoder for Features from Raw Sensor Data.

In this experiment, the raw sensor dataset was directly used to deploy machine learning models. So, each time stamp for every sensor is a feature and we don't extract process statistics like we did in the previous experiment. So, we have 85*654 = 55590 features and 298 observations. The raw sensor dataset was rescaled in the range of -1 to 1. As an initial experiment, we fixed the encoding dimension for our autoencoder same as we did in the previous experiment. We get the following graph shown below in Figure 4.3. Though from the graph we can see that there is no substantial improvement in MSE after 3000 neurons but due to memory constraints, we needed to further lower our encoding dimension for learning features by autoencoders from raw sensor dataset. We finally fixed 2000 neurons as our encoding dimension.

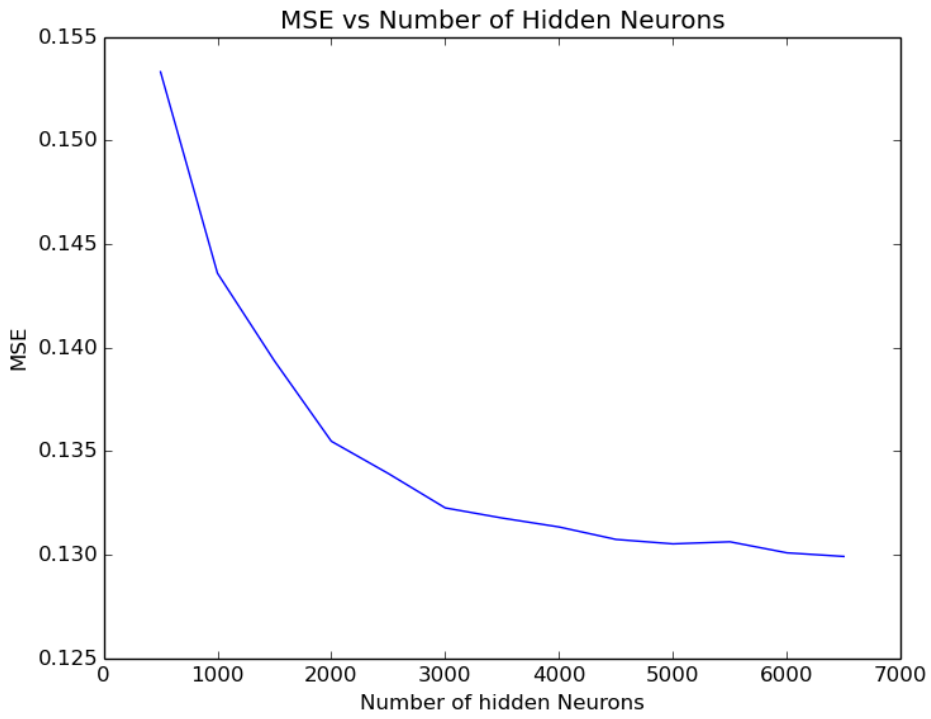

Figure 4.3 Graph of MSE and number of hidden neurons for raw sensor dataset.

We deploy prediction models on raw sensor dataset (RSD), on features learnt by one hidden layer autoencoder from raw sensor dataset (AE+RSD), on features learnt by L1 autoencoder from raw sensor dataset (L1-AE+RSD), on features learnt by L2 autoencoder from raw sensor dataset (L2-AE+RSD) and on features learnt by deep autoencoder from raw sensor dataset (DAE+RSD).

Table 4.6 Testing MSE for various prediction models

Models	LASSO	RIDGE	SVM	DNN
RSD	0.018793	0.026347	0.019961	0.042531
AE+RSD	0.018793	0.028658	0.019954	0.038543
L1-AE+RSD	0.018704	0.029371	0.019969	0.036850
L2-AE+RSD	0.019132	0.03328	0.019951	0.044230
DAE+RSD	0.018793	0.042783	0.019966	0.056706

Table 4.7 Statistics for top 30 Pearson correlation coefficients

Models	Pr_Mean	Pr_Max	Number of coefficients> 0.3
RSD	0.31410	0.613	18
AE+RSD	0.17434	0.223	0
L1-AE+RSD	0.17336	0.237	0
L2-AE+RSD	0.16920	0.211	0
DAE+RSD	0.17174	0.219	0

Table 4.8 Statistics for top 30 Spearman correlation coefficients

Models	Sr_Mean	Sr_Max	Number of coefficients> 0.3
RSD	0.34378	0.3566	237
AE+RSD	0.29834	0.4025	10
L1-AE+RSD	0.30627	0.3642	14
L2-AE+RSD	0.30180	0.3861	16
DAE+RSD	0.29508	0.3254	9

Table 4.6 shows that the features learned by one hidden layer autoencoder provide better performance for various prediction models. This can be attributed to correlation

coefficients also which is evident from Table 4.7 and Table 4.8. The interpretation of the correlation coefficients can be done in a similar way as done for section 4.1. We can clearly see that we have a comparable ratio of critical or useful features with autoencoder but in a low dimensional space which explains why we have better prediction performance with autoencoder learnt features.

So, it is evident from the above experiments that the features learned by autoencoder provide better prediction performance for most of the existing models. Another important conclusion is that learning features directly on the raw sensor dataset rather than extracting summary statistics provides better model performance since extracting statistics leads to losing the process information. So, we conclude from the results that learning new features through autoencoders improves the prediction accuracy of various existing models.

CHAPTER 5 Conclusion and Future Work

In this thesis, we proposed new deep learning based VM models for semiconductor manufacturing process which is effectively able to learn a better representation of inputs in a low dimensional feature space and improves the performance of various prediction models. The proposed model highlights the importance of learning better features by using neural network models which enhances the performance of existing VM modeling techniques. We used a 3 cross-validation for hyper-parameter tuning to get the best performing parameters for prediction models. To verify the effectiveness of our proposed VM model, we compared it with existing methods using real-life plasma-etching sensor data. Our experimental results have shown that the proposed VM model provides better results than existing VM techniques.

In future, we may develop our own customized autoencoder for this complex regression task. An autoencoder which can be specifically suited for learning better representations of functional data from raw sensor signals can be highly effective. One more issue which can be addressed in the future is increasing the robustness of our prediction model. VM modeling requires a method which along with estimating non-linear functional mapping between predictor and response variable also deals with outliers. The robust kernel-based regression method proposed by Hwang *et al*. (2015) can prove to be effective in VM applications for the reason stated above.

This thesis is the first of its kind approach in adopting the modern neural network models to learn better features for regression task of VM modeling. There is also a good amount of future research scope in adopting this proposed model and improving the performance of VM models by designing a new custom autoencoder as stated above

which can be suitable for functional data coming from the monitoring sensors. Further research with our proposed model can prove to be very effective for VM applications in semiconductor manufacturing and can make the process highly efficient.

References

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798-1828.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research , 13*, 281-305.

Chang J. Y. C. & Cheng, F. T. (2005). Application development of virtual metrology in semiconductor industry. In *Proceedings of the 31st Annual Conference IEEE Industrial Electronics Society*, 2005, (pp. 124–129). IEEE.

Chou, P. H., Wu, M. J., & Chen, K. K. (2010). Integrating support vector machine and genetic algorithm to implement dynamic wafer quality prediction system. *Expert System Applications*, 37(6), 4413-4424.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems, 2*, 303-314.

Ferreira, A., Roussy, A., & Condé, L. (2009). Virtual metrology models for predicting physical measurement in semiconductor manufacturing. In *Advanced Semiconductor Manufacturing Conference, 2009. ASMC'09. IEEE/SEMI* (pp. 149-154). IEEE.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* MIT Press.

Hinton, G. E., & R.Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science, 313*(5786), 504-507.

Hirai, T., & Kano, M. (2015). Adaptive virtual metrology design for semiconductor dry etching process through locally weighted partial least squares. *IEEE Transactions on Semiconductor Manufacturing*, *28*(2), 137-144.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*, 359-366.

Hung, M. H., Lin, T. H., Cheng, F. T., & Lin, R. C. (2007). A novel virtual metrology scheme for predicting CVD thickness in semiconductor manufacturing. *IEEE/ASME Transactions on Mechatronics*, *12*(3), 308-316.

Hwang, S., Jeong, M. K., & Yum, B. J. (2014). Robust relevance vector machine with variational inference for improving virtual metrology accuracy. *IEEE Transactions on Semiconductor Manufacturing*, *27*(1), 83-94.

Hwang, S., Kim, D., Jeong, M. K., & Yum, B.-J. (2015, August). Robust kernel-based regression with bounded influence for outliers. *Journal of the Operational Research Society*, *66*(8), 1385–1398.

Jia, F., Lei, Y., Lin, J., Zhou, X., & Lu, N. (2016). Deep neural networks: A promising tool for fault characteristic mining and intelligent diagnosis of rotating machinery with massive data. *Mechanical Systems and Signal Processing*, *72*, 303-315.

Khan, A. A., Moyne, J. R., & Tilbury, D. M. (2007). An approach for factory-wide control utilizing virtual metrology. *IEEE Transactions on Semiconductor Manufacturing*, *20*(4), 364-375.

Kingma, D. P., & Ba, J. L. (2015). ADAM: A method for stochastic optimization. *International Conference on Learning Representations, 2015.*

Kim, B., Jeong, Y. S., Tong, S., Chang, I., & Jeong, M. K. (2016). Step-down spatial

randomness test for detecting abnormalities in DRAM wafers with multiple spatial maps. *IEEE Transactions on Semiconductor Manufacturing*, 29(1), 57-65.

Kim, B., Jeong, Y. S., Tong, S., Chang, I., & Jeong, M. K. (2015). A regularized singular value decomposition-based approach for failure pattern classification on fail bit map in a DRAM wafer. *IEEE Transactions on Semiconductor Manufacturing*, 28(1), 41-49.

Kim, M., Kang, S., Lee, J., Cho, H., Cho, S. & Park, J. S. (2017). Virtual metrology for copper-clad laminate manufacturing. *Computers and Industrial Engineering.* 109, 280-287.

Ko, Y., Moon, P., Kim, C., Ham, M., Jeong, M. K., Garcia-Diaz, A., Myoung, J., & Yun, I. (2013). Predictive modeling and analysis of HfO2 thin film process based on Bayesian information criterion using PCA-based neural networks. *Surface and Interface Analysis*. 45(9), 1334-1339.

Kurz, D., De Luca, C. & Pilz, J. (2015). A sampling decision system for virtual metrology in semiconductor manufacturing. *IEEE Transactions on Auomation Science and Engneering.* 12(1), 75-83.

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, *10*(Jan), 1-40.

Lee, H., Kim, Y., & Kim, C. O. (2017). A deep learning model for robust wafer fault monitoring with sensor measurement noise. *IEEE Transactions on Semiconductor Manufacturing*, *30*(1), 23-31.

Lee, K., Kim, N., & Jeong, M. K. (2014). The sparse signomial classification and regression model. *Annals of Operations Research*, *216*(1), 257–286.

Lee, S. K., Kang, P., & Cho, S. (2014). Probabilistic local reconstruction for k-NN regression and its application to virtual metrology in semiconductor manufacturing. *Neurocomputing*, *131*, 427-439.

Lin, T.-H., Hung, M.-H., Lin, R.-C., & Cheng, F.-T. (2006). A virtual metrology scheme for predicting CVD thickness in semiconductor manufacturing. *IEEE International Conference on Robotics and Automation* (pp. 1054-1059). Orlando, Florida: IEEE.

Pampuri, S., Schirru, A., Fazio, G., & Nicolao, G. D. (2011, August). Multilevel lasso applied to virtual metrology in semiconductor manufacturing.. *IEEE International Conference on Automation Science and Engineering(CASE), 2011.* (pp. 244-249). IEEE.

Park, S., Jeong, S., Jang, Y., Ryu, S., Roh, H. J. & Kim G. H. (2015). Enhancement of the Virtual Metrology Performance for Plasma-Assisted Oxide Etching Processes by Using Plasma Information (PI) Parameters. *IEEE Transactions on Semiconductor Manufacturing.* 28 (3), 241-246.

Purwins, H., Nagi, A., Barak, B., Höckele, U., Kyek, A., Lenz, B., ... & Weinzierl, K. (2011). Regression methods for prediction of PECVD silicon nitride layer thickness. *IEEE International Conference on Automation Science and Engineering(CASE), 2011.* (pp. 387-392). IEEE.

Ruder, S. (2016). An overview of gradient descent optimisation algorithms. *arXiv preprint arXiv:1609.04747.*

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning internal representations by error propagation. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel Distributed Processing*, vol. 1 (pp. 318–362). MIT Press.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*, 1929-1958.

Susto, G. A., & Beghi, A. (2013). A virtual metrology system based on least angle regression and statistical clustering. *Applied Stochastic Models in Business and Industry*, *29*(4), 362-376.

Su, Y. C., Lin, T. H., Cheng, F. T., & Wu, W. M. (2008). Accuracy and real-time considerations for implementing various virtual metrology algorithms. *IEEE Transactions on Semiconductor Manufacturing*, 21(3), 426-434.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on Machine Learning* (pp. 1139-1147). Atlanta, Georgia, USA: PMLR.

Tran, V. T., AlThobiani, F., & Ball, A. (2014). An approach to fault diagnosis of reciprocating compressor valves using Teager–Kaiser energy operator and deep belief networks. *Expert Systems with Applications, 41*(9), 4113-4122.

Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory*.

Vincent, P., Larochelle, H., Bengio, Y. & Manzagol, P.A (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of 25th International Conference on Machine Learning (ICML)*. (pp. 1096–103). New York.

Zeng, D., & Spanos, C. J. (2009). Virtual metrology modeling for plasma etch operations. *IEEE Transactions on Semiconductor Manufacturing*, *22*(4), 419-431.