Journal of Process Control 84 (2019) 24–34

Contents lists available at ScienceDirect

Journal of Process Control

journal homepage: www.elsevier.com/locate/jprocont

DeepVM: A Deep Learning-based approach with automatic feature extraction for 2D input data Virtual Metrology

Marco Maggipinto[a], Alessandro Beghi[a,b], Seán McLoone[c], Gian Antonio Susto[a,b,*]

[a] Department of Information Engineering, University of Padova, Italy
[b] Human Inspired Technology Research Centre, University of Padova, Italy
[c] Centre for Intelligent Autonomous Manufacturing Systems, Queen's University Belfast, United Kingdom

ARTICLE INFO

Article history:
Received 20 March 2019
Revised 19 August 2019
Accepted 21 August 2019
Available online 16 October 2019

Keywords:
Advanced Process Control
Convolutional Autoencoder
Deep Learning
Etching
Feature extraction
Industry 4.0
Optical Emission Spectroscopy
Semiconductor Manufacturing
Soft sensor
Virtual Metrology

ABSTRACT

Industry 4.0 encapsulates methods, technologies, and procedures that transform data into informed decisions and added value in an industrial context. In this regard, technologies such as Virtual Metrology or Soft Sensing have gained much interest in the last two decades due to their ability to provide valuable knowledge for production purposes at limited added expense. However, these technologies have struggled to achieve wide-scale industrial adoption, largely due to the challenges associated with handling complex data structures and the feature extraction phase of model building. This phase is generally hand-engineered and based on specific domain knowledge, making it time consuming, difficult to automate, and prone to loss of information, thus ultimately limiting portability. Moreover, in the presence of complex data structures, such as 2-dimensional input data, there are no established procedures for feature extraction. In this paper, we present a Deep Learning approach for Virtual Metrology, called DeepVM, that exploits semi-supervised feature extraction based on Convolutional Autoencoders. The proposed approach is demonstrated using a real world Semiconductor Manufacturing dataset where the Virtual Metrology input data is 2-dimensional Optical Emission Spectrometry data. The feature extraction method is tested with different types of state-of-the-art autoencoder.

© 2019 Elsevier Ltd. All rights reserved.

1. Introduction

In recent years, industries have transitioned to collecting and archiving huge amounts of data from their production processes, leading to the so-called Big Data era. The challenges that Big Data pose in industrial environments are various [1] and the scientific community is in a continuous effort to propose innovative solutions to address them. One of the main problems in Industry 4.0 [2] is how to exploit the available data in order to obtain information that has business value. Increasingly, Machine Learning (ML) technologies that generate data-driven statistical inference models are being considered as a means of addressing this problem.

In Semiconductor Manufacturing, data-driven models play an important role in Advanced Process Control (APC) [3] where the complexity of the processes involved does not allow the creation of accurate physical models. Technologies such as Virtual Metrology (VM) [4–6], Predictive Maintenance [7], Fault Detection and Classification [8], and Yield Prediction [9] have grown in popularity, with consistent improvement in performance over time. In particular, VM, first proposed by Chen et al. [10] in 2005, is extensively applied in the semiconductor and other data-intensive manufacturing industries; its goal is to exploit the information already present in the system (eg. physical sensors measurements, tool settings) in order to infer the value of a costly or unmeasurable variable that is important for the operation of the production process or for characterizing the production quality. Usually this goal is achieved by means of *supervised learning* [11] methods where a Machine Learning model is created by leveraging labeled data where both the input and the output (the metrology prediction) have been physically measured from past process runs. In the Semiconductor Manufacturing (SM) literature, various VM solutions have been proposed for different output values such as thickness uniformity for Chemical Vapor Deposition [12], plasma electron density [13] and etch depth [14] for Plasma Etching, and removal rate for Chemical-Mechanical Planarization [15]. Beyond

* Corresponding author at: Department of Information Engineering, University of Padova, Italy.
E-mail addresses: marco.maggipinto@dei.unipd.it (M. Maggipinto), beghi@dei.unipd.it (A. Beghi), s.mcloone@qub.ac.uk (S. McLoone), gianantonio.susto@dei.unipd.it (G.A. Susto).

https://doi.org/10.1016/j.jprocont.2019.08.006
0959-1524/© 2019 Elsevier Ltd. All rights reserved.

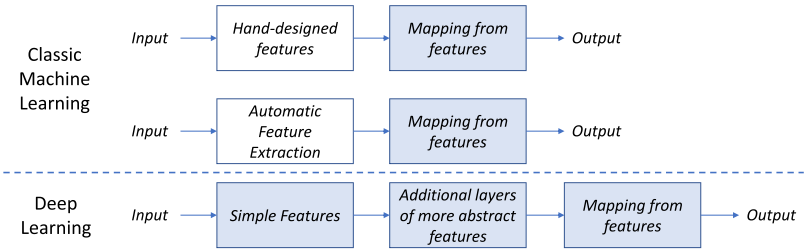


Fig. 1. Flowchart comparing the approaches of classical Machine Learning and Deep Learning in treating the feature extraction phase.

SM, VM technologies are now applied in many industries under different names, such as Soft Sensing [16] and Virtual Sensing [17].

In VM problems, the input data often exhibit complex structures. In particular, it is common to encounter data in the form of time series or with a multidimensional evolution, or both [16]. Traditional ML techniques that are usually employed in this context are not suitable for direct application to this sort of input data; rather, a preliminary operation called *feature extraction* is required where a set of informative values are extracted from the raw data and collected in a *design matrix* that can then be easily handled by traditional ML algorithms. The feature extraction phase can be performed in two ways (as depicted in Fig. 1):

- Hand-designed: the raw data is manually inspected to identify informative characteristics that can be represented as single parameter features in a design matrix. Input from equipment/process experts is usually required so that the process is guided by physical knowledge of the system under examination. Semi-automatic feature extraction methods [18,19] are also included in this category. Here subject matter expertise (SME) is incorporated in an automatic feature selection procedure in order to improve the quality of the extracted quantities.
- Automatic: automatic procedures are employed to extract potentially important features from the data. Such procedures are generally based on computing statistical properties on the input variables, sub-sampling, or averaging of different portions of the input data [20–22].

Both these approaches present significant drawbacks: hand-designed feature extraction is extremely time consuming since it requires a thorough graphical inspection of the data in order to understand which characteristics show variation that correlates with the prediction target. Moreover, it is poorly scalable in a complex environment like modern data-intensive and multi-stage manufacturing production and typically process specific since SME is included in the procedure. On the other hand, automatic feature extraction methods are typically not able to capture all the valuable information contained in the data leading to poor prediction capabilities. Recently, more sophisticated feature extraction methods have been proposed in order to overcome the aforementioned problem. In [23], a functional learning solution is presented that tackles feature extraction in a supervised fashion for time-series data embedding it in the modeling phase. In [24], an approach based on regularization [11] and Fused LASSO [25] is employed to deal with Optical Emission Spectroscopy (OES) data [14]; OES data are paradigmatic of the need for a sophisticated feature extraction mechanism due to their 2-dimensional evolution with respect to time and wavelength (as will be detailed in Section 4).

The 2-dimensional structure of OES data has characteristics similar to an image, suggesting the use of Computer Vision inspired methodologies. In particular, the convolution operation is highly effective at extracting local features from images. As a consequence, Convolutional Neural Networks are extensively employed for problems like object localization and recognition [26], face recognition [27], and text recognition [28]. For this reason, in this paper a VM approach, called *DeepVM*, that leverages an automatic feature procedure based on deep convolutional *autoencoders* is proposed. An autoencoder is a specific type of Artificial Neural Network (ANN) topology that is trained to reconstruct its input. Usually, the hidden layers of the network perform dimensionality reduction on the input, learning relevant features that allow satisfactory reconstruction. Moreover, deep autoencoders exploit multiple non-linear representational layers that learn complex hierarchical features from the data features that can be highly informative with regard to the underlying problem structure.

The main contributions of the present work are as follows:

- an exploration of the use of Convolutional Autoencoders in the field of Semiconductor Manufacturing;
- a comparison of various Autoencoder typologies presented in the literature but not previously employed in VM;
- a novel DeepVM multilayer feature extraction methodolgoy: in contrast to other approaches that employ Autoencoders for VM or soft sensing, and that only exploit the last layer of features extracted by the network, DeepVM leverages all the layers of the network, potentially proving a broader set of informative features;

Furthermore, to foster reproducibility of the results obtained in this work, the code and algorithm used to generate the results has been shared in a public repository.[1]

The remainder of the paper is organized as follows: Section 2 provides an overview of Deep Learning for feature extraction. DeepVM is introduced in Section 3. Autoencoder structures are also reviewed in this section. In Section 4 the Semiconductor Manufacturing case study and the experimental results are described. Final remarks and directions for future work are presented in Section 5.

2. Deep Learning for feature extraction

Deep Learning models have been applied to a wide variety of problems thanks to their inherent ability to treat complex input data without the need for time consuming and poorly scalable feature extraction procedures. Often, these methods are employed in a supervised fashion to solve the problem at hand. The high representational capabilities of DL make it a powerful automatic feature extraction method that can be used in combination with traditional ML techniques.

Feature embedding has a pivotal role in learning problems that deal with extremely complex data. In recent years, the diffusion of DL technologies has paved the way for sophisticated automatic feature extraction methods that are able to effectively compress the data in a lower dimensional representation without loss of information. The advancements in Computer Vision are the foundation of the incredible success of these technologies where DL based feature extraction mechanisms are widely employed: Romero et al. [29] employs a stacked Convolutional Autoencoder for unsupervised feature learning on remote sensing images. The unsupervised

[1] The code repository for the work described in this paper is available at the following link: https://gitlab.dei.unipd.it/dl_dei/DeepVM.

pre-training of the autoencoder, combined with a supervised fine-tuning, makes it possible to cope with the high-dimensionality of the data and the limited dataset size. The proposed method outperforms traditional methods such as Principal Component Analysis (PCA) and its kernel counterpart version, kPCA. Moreover, the deep architecture performs significantly better than shallow alternatives.

In [30] a Convolutional Neural network is used to extract informative features from hyperspectral images. The employed method is trained in a supervised way, layer-by-layer, for a classification problem. In particular, the task is to identify different regions (e.g. water, tree, asphalt) in satellite images.

Sun et al. [31] employs DL for face representation. The method relies on the use of CNNs to reduce the dimensionality of the significant regions of the input face, yielding a series of "DeepID" that collectively provides the input for a face verification model.

In the semiconductor industry, the diffusion of DL based automatic feature extraction methods is still in its infancy, however, some recent papers have employed them to solve complex problems in manufacturing, especially in the context of smart monitoring. Lin et al. [32] developed a single layer autoencoder for screening test escapes. In particular, the autoencoder is trained in an unsupervised fashion on non faulty chips only by using the Euclidean distance as a cost function. Then, the faulty chips are identified because of the higher reconstruction error. Lee et al. [33] designed a Denoising Autoencoder for wafer fault monitoring, showing the ability of the model to extract noise tolerant features from the data that led to high predictive capabilities. Two examples are also present in the literature for soft sensing tasks: In [34] the authors leverage a structure based again on Denoising Autoencoders to estimate oxygen levels in a coal-fired thermal power plant, while in [35] Deep Autoencoders are applied to a VM system for etching, in which time-series data are available as input. We remark, however, that no previous work in VM has adopted deep autoencoder-based solutions for OES or 2-dimensional input data.

In [16] we proposed a Deep Learning architecture for etch-rate prediction based on CNNs. DeepVM differs substantially from our previous work. In fact in [16] a CNN was trained from scratch to predict the etch rate; DeepVM instead exploits autoencoders trained on a reconstruction task in order to provide an automatic feature extraction method whose features are then fed to various regression algorithms. Since DeepVM is an automatic feature extractor, it also provides improved flexibility in the sense that its features can also be used for tasks other that etch rate prediction.

3. DeepVM

DeepVM consists of two main blocks (see Fig. 2), namely, a feature extraction module and a modeling (regression) one. While such blocks are typically present in VM solutions, the peculiarity of DeepVM is the feature extraction block that is based on a deep autoencoder and does not require hand-engineered procedures.

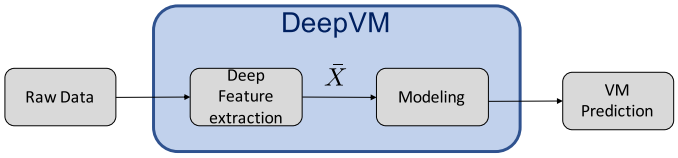


Fig. 2. The DeepVM architecture.

The proposed feature extraction method exploits the representational power of a CNN composed of three convolutional layers alternating with average pooling layers. The use of average pooling guarantees that smooth features are extracted, that are usually suitable for regression problems. Fig. 3 depicts in detail the proposed feature extraction procedure: The CNN is trained as described in Section 3; then, the features extracted by each average pooling layer are flattened and concatenated to form a final feature vector whose size is one-third of the original one.

The features from the autoencoder are fed to a modeling regression block that is trained in a supervised way to perform the VM target prediction.

We remark that the procedure is generic, that is, the deep autoencoder and the regression approach can be arbitrarily chosen. In this work we compare the performance of 'standard' [36], denoising [37] and variational [38] autoencoders for the feature extraction and of LASSO [11], Ridge Regression [11] and Support Vector Regression (SVR) [39] for the modeling.

For the sake of self-containedness, we devote the rest of this Section to providing a basic overview of Neural Networks and Autoencoders, referring interested readers to the literature for more detailed descriptions. Support Vector Regression is also briefly described, as it is the best performing of the regression algorithms investigated (as will be shown in Section 4) (Fig. 4).

3.1. Artificial Neural Networks

Artificial Neural Networks are the foundation of DL technologies. An ANN is the interconnection of simple units called *neurons* in a multilayer structure that emulates, in a rudimentary way, the human brain. We can distinguish three different types of network layer, namely, input, hidden, and output layers. The input layer provides the input values to the network. The output layer provides the output of the network and its structure is chosen to match the characteristics of the output; in particular, a *regression* function is employed when the output is continuous i.e. $y \in \mathbb{R}$, while a *classification* function (e.g. softmax) is used [11] when the output is categorical which means $y \in \{0, 1, \ldots, K\}$ where K is the number of classes. Hidden layers apply a transformation to the previous layer's output, with the transformation depending on the structure of the hidden layer itself.

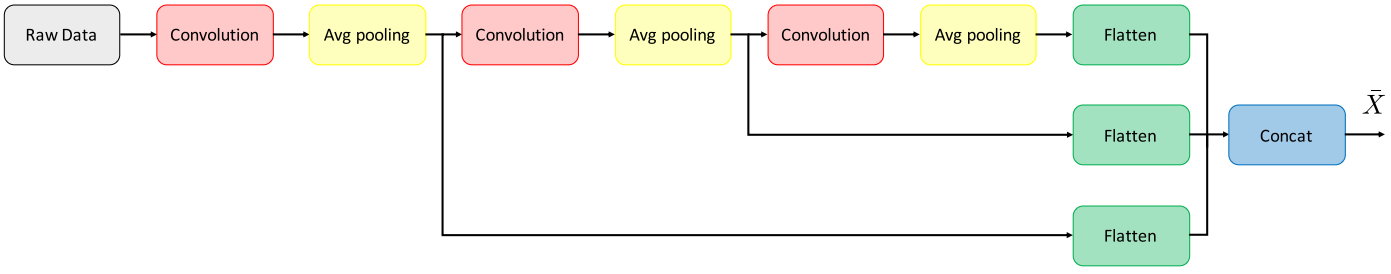


Fig. 3. DeepVM feature extraction.

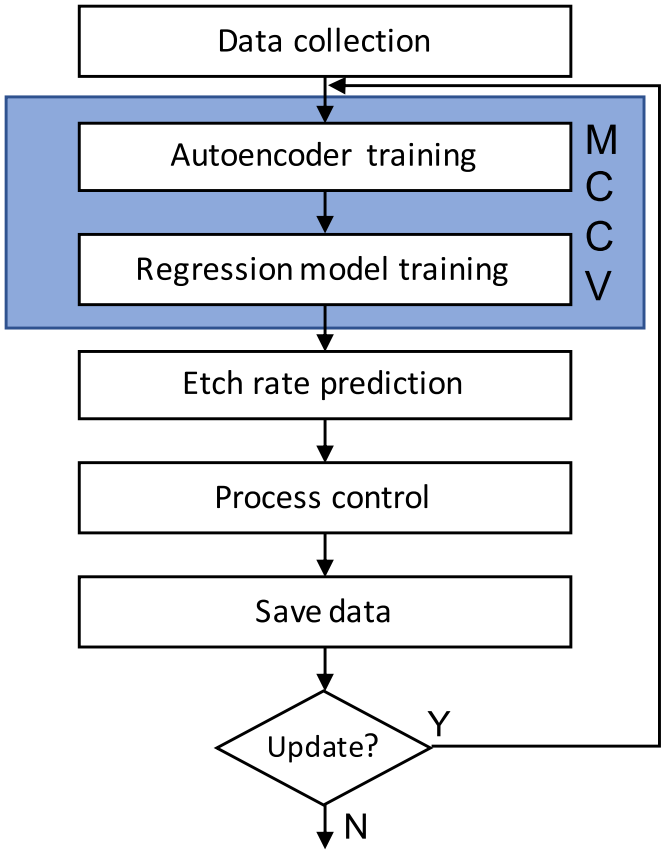


Fig. 4. DeepVM from development to production.

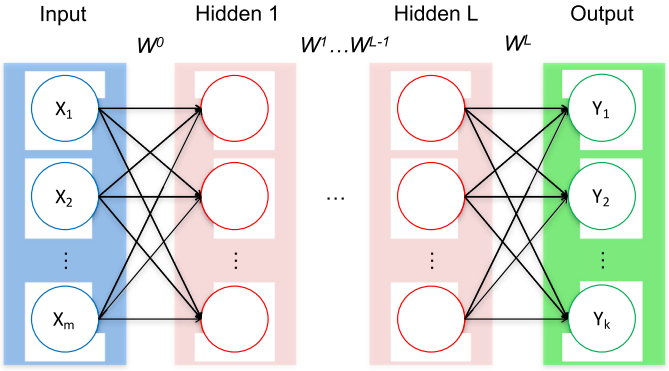


Fig. 5. Generic structure of a FNN. It is possible to distinguish the input layer (blue) output layer (green) and L hidden layers (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Various ANN structures have been developed over the years. The simplest one is the so-called Feedforward Neural Network (FNN) (Fig. 5) where the neurons are connected in a directed graph without any feedback loop. In this case, the hidden neurons apply a non-linear activation function σ to an affine transformation of the previous layer output. We can thus associate a matrix W^l and a bias vector b^l to each hidden layer l whose output can be computed as follows:

$$y^l = \sigma\left(W^l y^{l-1} + b^l\right) \tag{1}$$

Here $\sigma(\cdot)$ denotes element-wise application of the activation function σ. Given the number of neurons in lth layer q^l, and the number of neurons in layer $l-1$ q^{l-1}, the matrix W^l has size $q^l \times q^{l-1}$ and the bias vector b has length q^l. The output of the $(l-1)^{th}$ layer is a column vector of dimension q^{l-1} (Fig. 6).

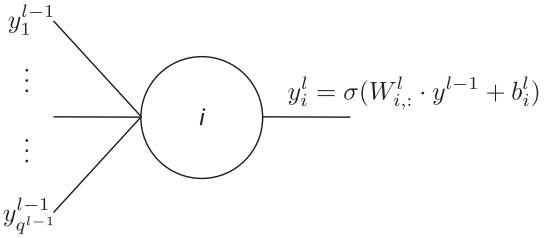


Fig. 6. Output of the ith neuron of the kth layer of a FNN. The notation $W_{i,:}^l$ indicates the ith row of the matrix W^l.

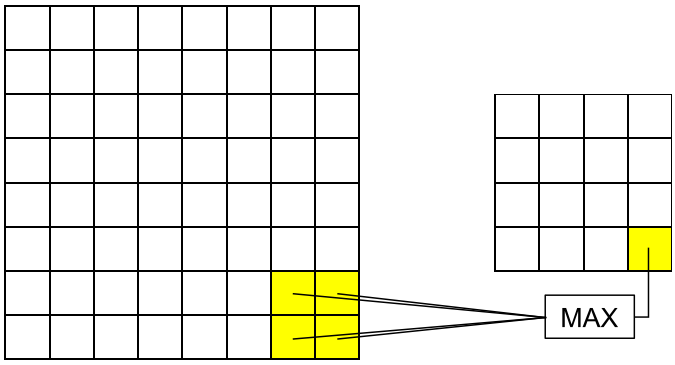


Fig. 7. Representation of the max pooling operation over a two-dimensional input using 2×2 regions.

Common choices for activation functions σ are the *sigmoid*, *tanh* and *rectifier linear unit (ReLu)* [40]. The *ReLu* function is often a good choice because of its similarity to a linear function and thanks to its constant gradient that does not vanish during training [36]. The ReLu function is defined as:

$$\sigma(x) = \max(0, x) \tag{2}$$

In recent years, more complex networks called Convolutional Neural Networks (CNNs) have gained popularity thanks to their performance in Computer Vision applications. CNNs exploit a multilayer structure similar to FNNs but with different types of hidden layers, that appear in an alternating fashion. In particular we can distinguish three kinds of hidden layer: (i) convolutional, (ii) pooling, (iii) fully connected.

(i) Convolutional layers are similar to the one employed in FNNs but in this case, each neuron applies the activation function to the convolution of the previous layer output with a kernel W^l plus a bias term b^l. The output of the lth convolutional layer can then be computed as follows:

$$y^l = \sigma\left(W^l * y^{l-1} + b^l\right) \tag{3}$$

Usually, multiple kernels are employed; therefore (3) is computed multiple times. Since the convolution operation can be applied in any dimension, it is common, in the presence of 2D data (images) to preserve the original input structure. The different outputs obtained using different kernels are then stacked and treated as *channels* (for further details see [36]). (ii) Pooling layers perform a subsampling of the previous layer output, usually by averaging (*average pooling*) or taking the maximum value (*max-pooling*) over a contiguous region of values. In Fig. 7 a graphical explanation of max-pooling is provided.

(iii) Fully Connected (FC) layers are the same as those employed in FNNs. Usually FC layers are placed at the end of the network. Since the structure of a FC layer is one-dimensional, multidimensional data are usually first *flattened* into a 1D vector.

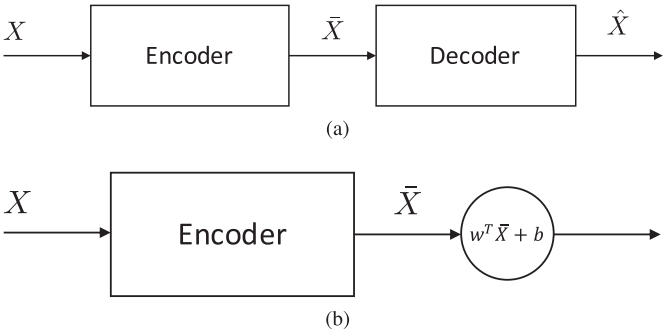


Fig. 8. (a) Structure of an autoencoder. X is the input, $\overline{X}$ is the compressed version of X and $\hat{\boldsymbol{X}}$ is the reconstructed version of X. (b) Encoder employed in the VM estimation. During *fine tuning* the pre-trained parameters of the encoder network are adjusted in a supervised way to achieve better performance.

ANNs provide an approximation function $y=f^*(x;\theta)$, that is parametrized by a set of coefficients θ (matrices and biases in FNNs, kernel/matrices and biases for CNNs) to an arbitrary complex continuous function f [36]. The creation of the predictive model thus requires the estimation of the parameters θ that best approximate the desired output. The estimation process consists of the minimization of a cost function defined according to the output layer properties. Common choices are Mean Squared Error for regression and cross-entropy for classification. Usually, a gradient-descent based algorithm is adopted, with the gradient computed using *backpropagation* [41].

3.2. Autoencoders

Autoencoders are composed of two main blocks: an encoder part that compresses the input into a low dimensional representation containing the informative content of the data; a decoder part that is trained to reconstruct the input from the features extracted by the encoder. Once the unsupervised pretraining is completed, the encoder part is thus a powerful automatic feature extractor that, augmented with a suitable output layer, can then be *fine-tuned* [36] in a supervised way to obtain the desired estimation performance.

Fine-tuning allows the parameters of the encoder to be adjusted to extract features that are the most effective in addressing the specific target estimation problem. This operation is performed by adding a further layer to the encoder block. Based on the problem at hand (See Fig. 8(b)), the encoder can be followed by either a classification or a regression output layer. In this latter case, a linear output layer is added to the encoder and the resulting network is trained in a supervised fashion. Note that previously computed encoder weights provide the starting point for the stochastic gradient descent algorithm. As a consequence, fine tuning is expected to refine the features extracted in an unsupervised fashion.

Autoencoders can be created using various Neural Network structures. In this paper we propose a Convolutional Autoencoder where the underlying ANN exhibits a convolutional structure as described in Section 3.1. Various kinds of autoencoder have been proposed in the literature, as briefly described in the following subsubsections.

3.2.1. Standard autoencoders

Standard autoencoders are simply trained to reconstruct the provided input by employing the encoder-decoder structure described above. In the following we will refer to standard autoencoders as 'autoencoders'.

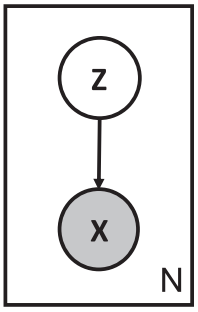


Fig. 9. Bayesian network describing a VAE.

3.2.2. Denoising Autoencoders

A Denoising Autoencoder (DAE) has the same structure as a standard autoencoder but it uses an augmented version of the original input where Gaussian noise has been added. The model is then trained to reconstruct the original input removing the noise. In this way, a set of features that effectively characterize the structure of the data and are not affected by the presence of noise is typically obtained. For more details on DAEs we refer interested readers to [37].

3.2.3. Variational Autoencoders

Variational Autoencoders (VAEs) have been proposed in [38] as a generative model that learns a model of the data distribution to generate new samples from it. Their auto-encoding structure makes them appealing from a feature extraction perspective, since they provide an embedded representation of the input. The idea behind VAEs is to implement a probabilistic model described by the Bayesian network of Fig. 9 where $\mathbf{X}$ represents the data and $\mathbf{Z}$ is a latent vector that is not available in the dataset. The joint probability density induced by the network of Fig. 9 is $p(\mathbf{x},\ \mathbf{z})=p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ where $p(\boldsymbol{z}\ )$ is a prior distribution, typically multivariate Gaussian. Training the model by maximum likelihood would require marginalizing out the latent variables $\mathbf{z}$ but this is not feasible due to the size of $\mathbf{z}$, hence, a variational approximation is made, introducing an approximate posterior $q(\mathbf{z}|\mathbf{x})$. In particular, in a VAE, both $p(\mathbf{x}|\mathbf{z})$ and $q(\mathbf{z}|\mathbf{x})$ are modeled with neural networks parametrized by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. The model is then trained by maximizing the variational lower bound [38]:

$$\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) = E_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - D_{\mathrm{KL}}\big(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}))\ \big\|\ p(\boldsymbol{z})\big) \tag{4}$$

For more details on VAEs we refer interested readers to [42].

3.3. Regularization

Linear regression is a well known prediction algorithm that assumes a linear input–output relationship $\boldsymbol{y}=\boldsymbol{\theta}^T\boldsymbol{x}$ parametrized by the vector $\boldsymbol{\theta}$ that is learned during training in order to minimize the prediction error, usually measured in terms of the MSE, on a training set. This simple method is however prone to overfitting when the number of features is high with respect to the number of training samples or in the presence of collinearity between features. For this reason, regularization approaches are employed where a penalty on the parameters vector norm is introduced in the cost function reducing the model complexity and biasing it towards simpler functions. The most common regularization approaches are *Ridge regression*, which introduces the L_2 norm of the parameters $\|\boldsymbol{\theta}\|_2^2$ as the penalty term, and *LASSO*, which employs the L_1 norm of the parameters $\|\boldsymbol{\theta}\|_1$ as the penalty term. LASSO also has the attractive property of inducing sparsity in the solution. Over the years, these methods have been extended and adapted to different use cases. Of particular interest is the Fused LASSO

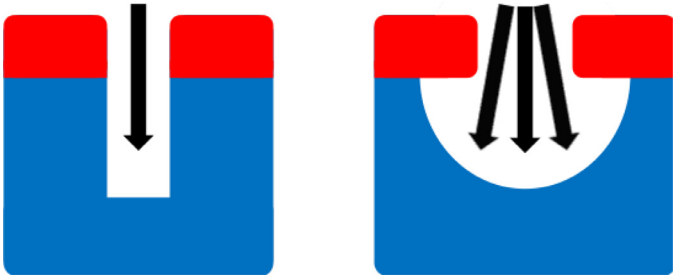


Fig. 10. Comparison between Directional Etching (left) and Isotropic Etching (right).

[25] which provides an effective approach to dealing with data that exhibit a temporal evolution. Specifically, a penalty on the difference of consecutive coefficients $\sum_{t=1}^{T} \| \theta_{t-1} - \theta_t \|_2^2$ is added to the normal LASSO cost function. This generates a sparse model thanks to the LASSO regularization while encouraging the coefficients for consecutive time instants to be "similar". This is a desirable property since it promotes selection of entire portions of the time series that are relevant for the prediction task while discarding the others, making the final model more interpretable. The same does not apply for the standard LASSO which instead treats different time instants independently without taking into account their temporal evolution.

3.4. Support Vector Regression

Support Vector Regression aims at finding a hyperplane $y = < w, x > + b$ such that the prediction error on the output variable y is less then a predefined constant ϵ. To obtain a smooth function, the norm of the parameters w is required to be small. This problem can be expressed as a convex optimization task:

$$\begin{aligned} &\min \quad \frac{1}{2} \| w \|^2 \\ &\text{subject to} \quad \begin{cases} y_i - < w, x_i > - b \leq \epsilon \\ < w, x_i > + b - y_i \leq \epsilon \end{cases} \end{aligned} \tag{5}$$

To make the solution feasible, a set of slack variables ξ_i, ξ_i^* are included in the problem resulting in:

$$\begin{aligned} &\min \quad \frac{1}{2} \| w \|^2 \\ &\text{subject to} \quad \begin{cases} y_i - < w, x_i > - b \leq \epsilon + \xi_i \\ < w, x_i > + b - y_i \leq \epsilon + \xi_i^* \end{cases} \end{aligned} \tag{6}$$

This problem is then solved by using Lagrangian multipliers to give an effective linear regression method, that can also be extended to non-linear problems by using non-linear kernels [39].

4. Experimental results

4.1. Semiconductor Manufacturing case study

Etching is a key step in the realization of integrated circuits, in which a masked silicon wafer is hit by a high-speed stream of plasma of an appropriate ionized gas mixture in a vacuum chamber. The exposed surface is thus etched away because of the chemical and mechanical stress released in the collision. In modern manufacturing, the majority of etching processes require directional etching, where the material is etched perpendicularly to the wafer surface (Fig. 10). This is achieved by accelerating the ions with a voltage bias [43]. Plasma etching processes suffer from the influence of various factors that may alter the final quality of the product, in particular, chamber mismatch, non-uniformity across the wafer and within die, and surface composition/roughness [44].

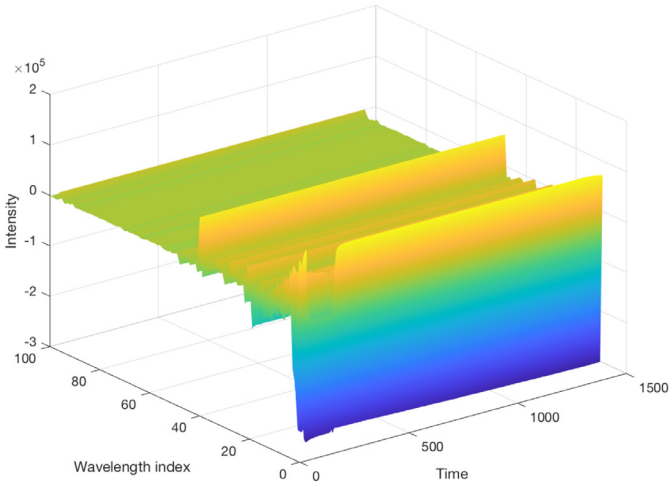


Fig. 11. An example of OES data during the Etching process.

For this reason, it is important for process control and quality assessment to understand the resolution and directionality of etching. In this regard, the *etch rate*, i.e. the thickness of the eroded surface per unit time, provides important information. However, measuring the etch rate requires a post-processing metrology step that is both time-consuming and extremely costly. It is thus pivotal for cost reduction and production performance to estimate the etch rate from cheap and easy to obtain measurements. To this end, Optical Emission Spectroscopy (OES) sensing of the plasma can be used to observe changes in the plasma chemistry during etching, thus providing the foundation for VM solutions that exploit historical data to build predictive models for etch rate estimation.

We propose using the DeepVM algorithm described in the previous sections to build such a VM solution. In particular, different structures corresponding to different choices of the autoencoder and the regression module will be analyzed and their performance compared using a case study provided by an industrial partner involved in the manufacture of storage media. The case study dataset consists of OES spectra and associated etch rate values for $N = 1554$ wafers processed through a single etch chamber. The OES data, which serves as the VM model input, has a complex 2-dimensional structure, with time and wavelength evolution, as depicted in Figs. 11 and 12. The 2-dimensional structure of OES suggests the use of Computer Vision inspired technologies, thus motivating the use of models based on CNNs, that have outperformed other methods in many Computer Vision tasks [45]. CNNs are able to extract hierarchical sparse features [36] from complex data like images. As such, the proposed method is expected to provide a powerful feature extraction model for OES data.

4.2. Experimental settings

Since, to the best of our knowledge, there are no publicly available datasets for comparing Deep Learning based VM approaches in Semiconductor Manufacturing, it is difficult to define the state-of-the-art. Consequently, to asses the quality of the proposed procedure, DeepVM will be compared with popular VM approaches that exploit simple feature extraction procedures. Also, a comparison is proposed with a recent approach for VM with OES data based on Fused LASSO [24].

The simple automatic feature extraction approach exploited for comparison can be summarized as follows: (i) a set of statistics is defined (mean, variance, skewness, kurtosis, maximum and mini-

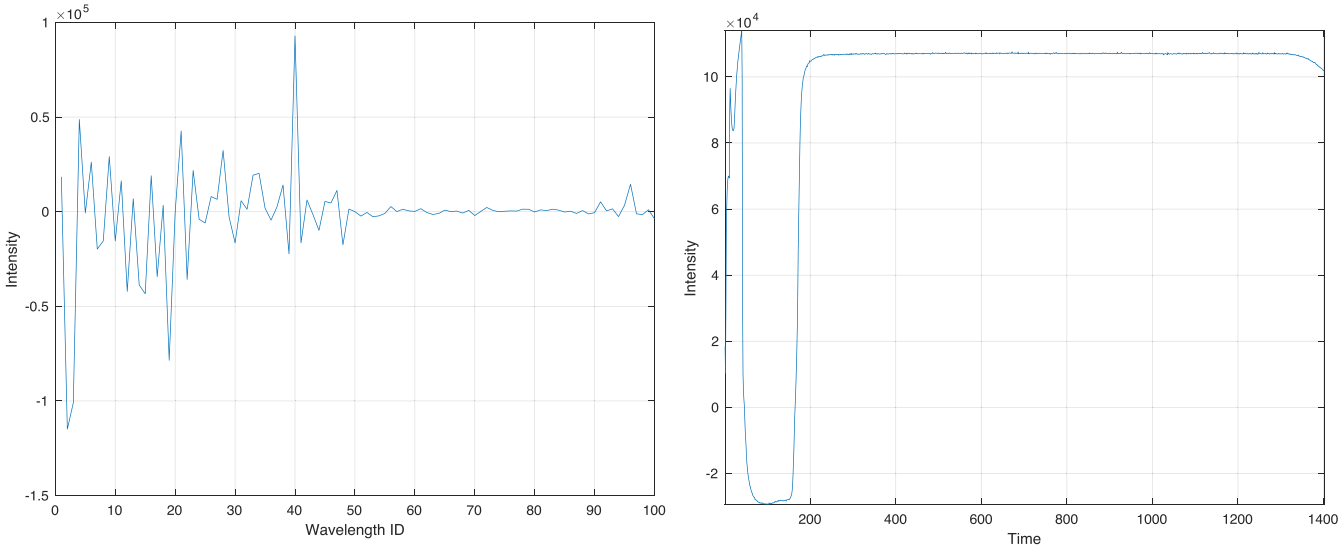


Fig. 12. An example of OES data during the Etching process for a fixed time sample (left) and a fixed wavelength (right).

mum value); (ii) these statistics are computed over the time evolution of each wavelength of the OES data. The resulting 'simple features' are then used in our experiments as inputs to the LASSO and Ridge Regression models. The approach based on Fused LASSO proposed in [24] can be divided in the following main steps: (i) a dimensionality reduction of the wavelengths is performed and the 'most informative' wavelengths selected; (ii) Fused LASSO is performed on the retained wavelengths. We refer interested readers to [24] for details of the procedure.

The training and development of the proposed algorithm has been realized using Keras [46] with Tensorflow [47]. An *adam* optimizer has been employed using Mean Squared Error (MSE) as the cost function. Specifically, the training operation has been performed in two steps: unsupervised pre-training, where the MSE measures the reconstruction error of the AE followed by supervised fine-tuning where the MSE measures the prediction error of the model for the targets y. It is worth remarking that a fine-tuning procedure (as discussed in Section 3.2) is employed to obtain more representative features, while the final prediction algorithm is realized using a SVR that takes as input the concatenation of all the features coming from the pooling layers of the encoder as depicted in Fig. 3.

The performance of the proposed method has been assessed using 20 Monte-Carlo cross validation (MCCV) [48] cycles with a test set composed of 30% of the total number of process runs available, i.e. $N_{test} = 0.3 \cdot N$ and $N_{train} = 0.7 \cdot N = N - N_{test}$. The same procedure has been employed to estimate the performance of the benchmark methods.

To estimate the hyperparameters of the employed regression algorithms, 5 Monte-Carlo cross validation cycles have been performed on a validation set composed of 30% of the available training data, i.e. $N_{val\_t\ est} = 0.3 \cdot N_{train}$.

The metrics employed to quantify model prediction capabilities are the MSE and R^2 score.

4.3. Results

The mean and standard deviation of the performance indexes computed over 20 MC cross validation cycles are reported in Table 1. DeepVM with 'standard' autoencoder plus SVR is the best approach both in terms of R^2 and MSE. Moreover, it can be seen that irrespective of the type of autoencoder employed, DeepVM provides at least one solution that outperforms the other approaches. These two considerations confirm the ability of Deep Learning methods to provide good feature extraction capabilities for VM. The lower performance provided by VAEs may be explained by the generative nature of such models that typically require more data to be trained. Moreover, the KL term in the lower bound tends to limit the capacity of the encoder; this effect could be reduced by adding a hyperparameter that tunes the capacity of the model. However, such an approach has not been explored in this paper since validating the added hyperparameter would be extremely expensive in terms of computational resources. A more detailed performance comparison is reported in Fig. 13. This shows the boxplots of the distribution of the performance indexes for each method over 20 MC cross validation cycles. In Fig. 14 we plot the predicted etch rates against the true values for the proposed DeepVM and the methods based on statistical features. It is noticeable how the points are all around the $y = x$ line, meaning that the predictions provide useful information about the real value of the etch-rate. Of course, the methods based on statistical features have more dispersed scatter plots, reflecting the inferior performance of these methods, as observed in the boxplots and tables.

In Tables 3–5 we report the performance of DeepVM when only the features coming from the last two layers of the feature extraction module are used to perform the prediction. Such a strategy may be desirable to reduce the prediction time and model complexity. A performance drop can be observed with all the modeling techniques, however, DeepVM with AE and SVR continues to have superior performance to the other methods. It is thus possible to define a trade-off between prediction accuracy and complexity, based on the automatic feature selection method. We remark that in real industrial environments, it may be important to reduce the time required to compute the VM prediction, in particular when they are used for control purposes [49]. In Table 2 we report the percentage of predictions with an error within 5 and 10% of the real value. It is noticeable that for the 10% case all the methods achieve results close to 99% while for the 5% case DeepVM has substantially better performance that, for the application at hand, are considered acceptable for real use. It is important to remark that the performance of our model is achieved without including 'subject matter expertise' in the process, hence the method can be easily adapted to different processes/machines. Furthermore, the etch rate predictions obtained can be generated for every wafer during production, whereas physical measurements are usually

Table 1
Performance comparison of the considered VM approaches: the best performances are reported in **bold**. Results are averaged over 20 MCCV cycles and reported in the format "mean $\pm$ 1 $\cdot$ std".

	Conv Net	LASSO	Fused LASSO	DeepVM AE + SVR	DeepVM AE + LASSO	DeepVM AE + Ridge	DeepVM DAE + SVR	DeepVM DAE + LASSO	DeepVM DAE + Ridge	DeepVM VAE + SVR	DeepVM VAE + LASSO	DeepVM VAE + Ridge
R^2	0.36 ± 0.07	0.42 ± 0.04	0.39 ± 0.01	$\mathbf{0.52 \pm 0.08}$	0.44 ± 0.13	0.51 ± 0.13	0.48 ± 0.15	0.43 ± 0.12	0.49 ± 0.16	0.42 ± 0.11	0.40 ± 0.09	0.46 ± 0.13
MSE [10^{-5}]	2.78 ± 0.52	2.66 ± 0.36	2.59 ± 0.56	$\mathbf{2.34 \pm 0.56}$	2.74 ± 0.76	2.39 ± 0.76	2.44 ± 0.93	2.65 ± 0.73	2.37 ± 0.95	2.64 ± 0.57	2.71 ± 0.53	2.44 ± 0.58

Table 2
Percentage of predictions with an error less then 10% and 5% of the real value.

	SVR	LASSO	Ridge	DeepVM AE + SVR	DeepVM AE + LASSO	DeepVM AE + Ridge	DeepVM DAE + SVR	DeepVM DAE + LASSO	DeepVM DAE + Ridge	DeepVM VAE + SVR	DeepVM VAE + LASSO	DeepVM VAE + Ridge
$\% \pm 10\%$	**99.36**	92.07	99.35	98.93	99.14	98.72	98.93	**99.36**	98.93	98.50	98.71	98.71
$\% \pm 5\%$	82.65	63.59	83.94	89.29	87.15	90.57	88.86	85.22	88.65	88.44	86.73	**91.01**

Table 3
Performance comparison of DeepVM and AE for various feature selection (all or only the features from the last two layers) and modeling approaches (SVR, LASSO and Ridge). Results are averaged over 20 MCCV cycles and reported in the format "mean $\pm$ 1 $\cdot$ std".

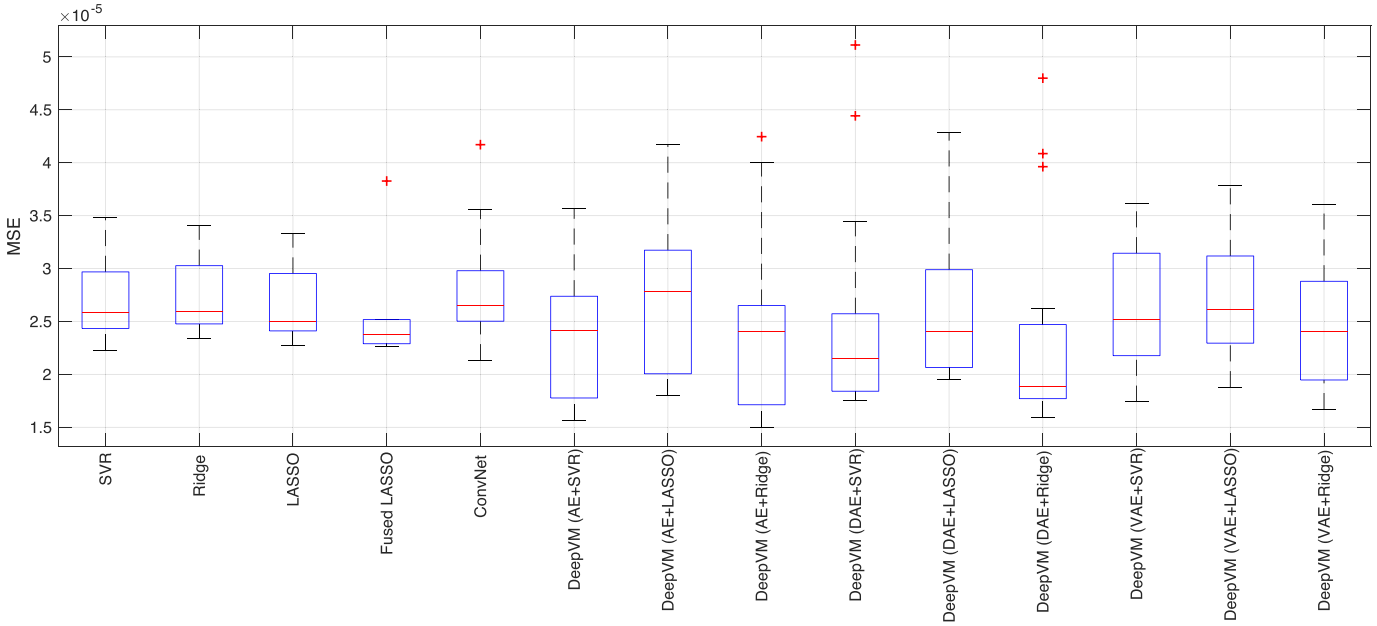
	AE (All Layers) + SVR	AE (Last 2 Layers) + SVR	AE (All Layers) + LASSO	AE (Last 2 Layers) + LASSO	AE (All Layers) + Ridge	AE (Last 2 Layers) + Ridge
R^2	$\mathbf{0.52 \pm 0.08}$	0.41 ± 0.11	$\mathbf{0.44 \pm 0.13}$	0.42 ± 0.09	$\mathbf{0.51 \pm 0.13}$	0.44 ± 0.12
MSE [10^{-5}]	$\mathbf{2.34} \pm 0.56$	2.83 ± 0.56	$\mathbf{2.74 \pm 0.77}$	2.83 ± 0.57	$\mathbf{2.39 \pm 0.76}$	2.71 ± 0.61

Table 4
Performance comparison of DeepVM and DAE for various feature selection (all or only the features from the last two layers) and modeling approaches (SVR, LASSO and Ridge). Results are averaged over 20 MCCV cycles and reported in the format "mean $\pm$ 1 $\cdot$ std".

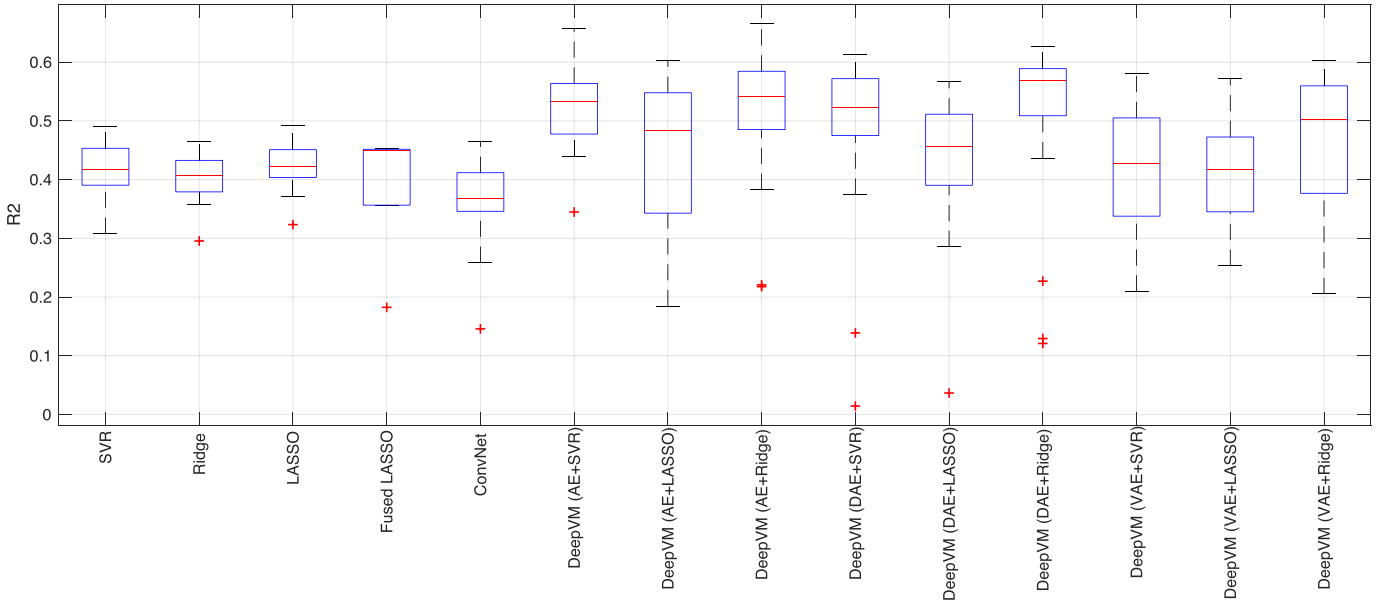
	DAE (All Layers) + SVR	DAE (Last 2 Layers) + SVR	DAE (All Layers) + LASSO	DAE (Last 2 Layers) + LASSO	DAE (All Layers) + Ridge	DAE (Last 2 Layers) + Ridge
R^2	$\mathbf{0.48 \pm 0.15}$	0.43 ± 0.10	$\mathbf{0.43 \pm 0.12}$	0.42 ± 0.09	$\mathbf{0.49 \pm 0.16}$	0.45 ± 0.09
MSE [10^{-5}]	$\mathbf{2.44 \pm 0.93}$	2.68 ± 0.73	$\mathbf{2.65 \pm 0.73}$	2.69 ± 0.69	$\mathbf{2.37 \pm 0.95}$	2.52 ± 0.49

Table 5
Performance comparison of DeepVM and VAE for various feature selection (all or only the features from the last two layers) and modeling approaches (SVR, LASSO and Ridge). Results are averaged over 20 MCCV cycles and reported in the format "mean $\pm$ 1 $\cdot$ std".

	VAE (All Layers) + SVR	VAE (Last 2 Layers) + SVR	VAE (All Layers) + LASSO	VAE (Last 2 Layers) + LASSO	VAE (All Layers) + Ridge	VAE (Last 2 Layers) + Ridge
R^2	$\mathbf{0.42 \pm 0.11}$	0.39 ± 0.09	$\mathbf{0.40 \pm 0.08}$	0.40 ± 0.08	$\mathbf{0.46 \pm 0.13}$	0.41 ± 0.08
MSE [10^{-5}]	$\mathbf{2.64 \pm 0.57}$	2.77 ± 0.54	$\mathbf{2.71 \pm 0.53}$	2.74 ± 0.49	$\mathbf{2.44 \pm 0.58}$	2.71 ± 0.54



(a) MSE



(b) R^2 score

Fig. 13. Prediction performance on the OES etching dataset over 20 MC cross validation cycles.

taken only on a per-lot basis. It has been shown that controller performance can be greatly improved by exploiting these estimates [50,51]. The low values obtained for the R^2 metric are justified by the complexity of the process at hand. This is a multi-step production process [52] and what we are addressing here is an early stage prediction i.e. we are trying to predict the etch rate for the first step of the process while the actual metrology measurement is performed at the end of the multi-step process. Hence the measured value incorporates variations introduced by the other steps that cannot be predicted by the OES data recorded for the first step.

In Table 6 we report the execution time for the statistical feature extraction method and the DeepVM approach. The evaluation time on a single CPU is 6 times greater with the Deep Learning based approach. However, the advantage of the DL methods is that they can be easily parallelized on GPUs. In this case, the execution time of DeepVM improves by a factor of 10 (3.5 ms versus 37 ms). In practical terms, both execution times are sufficiently fast to have no impact on performance in an etch chamber run-to-run control scenario, hence the proposed method is suitable for deployment

Table 6
Exection time of the different feature extraction methods.

	Stat features	DeepVM CPU	DeepVM GPU
Execution time [ms]	6.67 ± 0.02	36.97 ± 0.33	3.52 ± 0.18

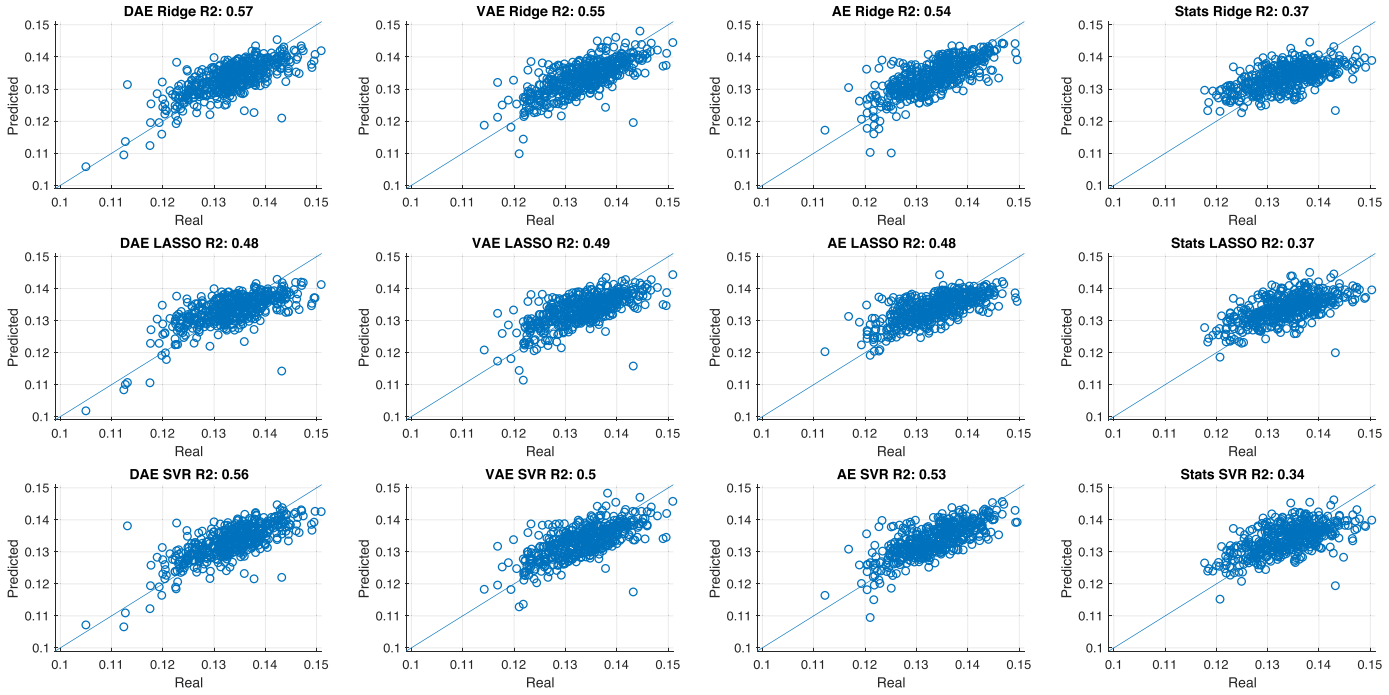


Fig. 14. Scatter plot of the predicted values of the etch rate on the test set.

in production. Training of the DeepVM model is a computationally demanding exercise, but this can be completed off-line. The the full cross validated training of the DeepVM model takes approximately 23 h on a machine with a single Titan Xp GPU and an Intel Core i7-6800K CPU.

A common problem in industrial environments is keeping VM models up-to-date in the presence of process drifts that may compromise their prediction capabilities. Deep Learning models are able to learn increasingly complex features of the data thanks to their multilayer structure. This means that, at least the first layers of the architecture, tend to extract general features (e.g. edges in images) that may still remain valid in the presence of changes in the distribution of the input. Thanks to this property, re-training the model periodically is a feasible approach to coping with distribution drifts, because it typically takes much less time to train a deep neural network when its weights have been initialized on a similar dataset. Furthermore, as process drift phenomena usually develop slowly, a model updating frequency of once per day is likely to be adequate.

As a final remark, we note that Ridge Regression consistently outperforms LASSO for this case study; this is usually associated with a high level of collinearity among the features, an issue that can be mitigated by preprocessing with a feature selection method such as [53].

5. Conclusion

DeepVM is an approach to Virtual Metrology that exploits an automated feature extraction method based on convolutional autencoders. Combined with traditional Machine Learning algorithms, DeepVM is able to effectively deal with the data complexity typical of the semiconductor industry, as shown by its application to the design of a VM module for etch rate estimation from OES data. In contrast to traditional ML algorithms that require the input to be organized in a design matrix where each row represents a single data observation, DeepVM can be applied in scenarios where each observation is a matrix itself with a 2-dimensional evolution.

DeepVM (standard AE and DAE implementations) outperforms classical shallow regression technologies, providing an accurate prediction of the required target (etch rate). The level of performance justifies the use of a complex DL model that, by exploiting the representational power of CNNs, is able to deal with the inherent 2-dimensional interdependence of OES data that exhibit both time and wavelength evolution. The proposed method presents considerable advantages over methods using hand crafted features, since it does not require any domain specific knowledge and is able to treat the input complexity in a natural and scalable way. Furthermore, the proposed solution is well suited to the Big Data context where historical data is in continuous growth, since a characteristic of DL algorithms is that their performance improves with increasing data availability [36].

We remark that the proposed DeepVM approach is intended for input data that exhibit complex 2-dimensional structure (such as images) and is likely not to offer any advantages over ‘shallow’ Machine Learning approaches on more conventional tabular process data. When both types of data are available, our method can easily be extended by concatenating the features coming from the autoencoder with the tabular data before feeding them to the regression algorithm at the end of the pipeline. It is worth highlighting that, whether SME can be included in the feature selection process, the performace of our model and all the others can be improved. However, SME was not available in this work and the proposed model has more general applicability since it can be trained also on data from a different process provided that they exhibit a similar structure. As a byproduct of the proposed approach, the autoencoders can also be exploited for compression purposes to optimize storage resources for tasks other than Virtual Metrology (e.g. quality monitoring, anomaly detection and smart monitoring). We remark that this capability does not exist with conventional feature extraction methods. Future work will seek to quantify this additional benefit. In addition, other feature selection approaches such as FSCA [53] will be investigated. These have the potential to improve prediction performance and may be useful for optimizing the trade-off between prediction performance and execution time.

Conflict of interest

The authors declare no conflicts of interest.

Acknowledgements

We would like to thank Nvidia Corporation for supporting our work with a Nvidia Titan V GPU.

References

[1] J. Moyne, J. Samantaray, M. Armacost, Big data capabilities applied to semiconductor manufacturing advanced process control, IEEE Trans. Semicond. Manuf. 29 (4) (2016) 283–291.
[2] J. Lee, H.-A. Kao, S. Yang, Service innovation and smart analytics for industry 4.0 and big data environment, Proc. CIRP 16 (2014) 3–8 http://www.sciencedirect.com/science/article/pii/S2212827114000857, doi:10.1016/j.procir.2014.02.001.
[3] S.-K.S. Fan, Y.-J. Chang, An integrated advanced process control framework using run-to-run control, virtual metrology and fault detection, J. Process Control 23 (7) (2013) 933–942.
[4] R. Langone, C. Alzate, B. De Ketelaere, J. Vlasselaer, W. Meert, J.A. Suykens, Ls-svm based spectral clustering and regression for predicting maintenance of industrial machines, Eng. Appl. Artif. Intell. 37 (2015) 268–278.
[5] S. Kang, P. Kang, An intelligent virtual metrology system with adaptive update for semiconductor manufacturing, J. Process Control 52 (2017) 66–74.
[6] M. Terzi, C. Masiero, A. Beghi, M. Maggipinto, G.A. Susto, Deep learning for virtual metrology: modeling with optical emission spectroscopy data, in: 2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI), IEEE, 2017, pp. 1–6.
[7] C.-Y. Lee, T.-S. Huang, M.-K. Liu, C.-Y. Lan, Data science for vibration heteroscedasticity and predictive maintenance of rotary bearings, Energies 12 (5) (2019) 801.
[8] G.A. Susto, A. Schirru, S. Pampuri, A. Beghi, G. De Nicolao, A hidden-gamma model-based filtering and prediction approach for monotonic health factors in manufacturing, Control Eng. Pract. 74 (2018) 84–94.
[9] J. Moyne, B. Schulze, Yield management enhanced advanced process control system (ymeapc) – Part I: Description and case study of feedback for optimized multiprocess control, IEEE Trans. Semicond. Manuf. 23 (2) (2010) 221–235.
[10] P. Chen, S. Wu, J. Lin, F. Ko, H. Lo, J. Wang, C. Yu, M. Liang, Virtual metrology: a solution for wafer to wafer advanced process control, ISSM 2005, IEEE International Symposium on Semiconductor Manufacturing, 2005, 2005, pp. 155–157.
[11] J. Friedman, T. Hastie, R. Tibshirani, The Elements of Statistical Learning, vol. 1, Springer Series in Statistics Springer, Berlin, 2001.
[12] G.A. Susto, A. Beghi, Least angle regression for semiconductor manufacturing modeling, 2012 IEEE International Conference on Control Applications (CCA), 2012, pp. 658–663.
[13] S.A. Lynn, N. MacGearailt, J.V. Ringwood, Real-time virtual metrology and control for plasma etch, J. Process Control 22 (4) (2012) 666–676.
[14] L. Puggini, S. McLoone, Extreme learning machines for virtual metrology and etch rate prediction, Signals and Systems Conference (ISSC), 2015 26th Irish, 2015, pp. 1–6.
[15] X. Jia, Y. Di, J. Feng, Q. Yang, H. Dai, J. Lee, Adaptive virtual metrology for semiconductor chemical mechanical planarization process using gmdh-type polynomial neural networks, J. Process Control 62 (2018) 44–54.
[16] M. Maggipinto, M. Terzi, C. Masiero, A. Beghi, G.A. Susto, A computer vision-inspired deep learning architecture for virtual metrology modeling with 2-dimensional data, IEEE Trans. Semicond. Manuf. 31 (3) (2018) 376–384.
[17] P.H. Ibargüengoytia, M.A. Delgadillo, U.A. García, A. Reyes, Viscosity virtual sensor to control combustion in fossil fuel power plants, Eng. Appl. Artif. Intell. 26 (9) (2013) 2153–2163.
[18] C. Jin, Methodology on Exact Extraction of Time Series Features for Robust Prognostics and Health Monitoring, Ph.D. Thesis, University of Cincinnati, 2017.
[19] J.A. Moyne, B. Schulze, J. Iskandar, M. Armacost, Next generation advanced process control: leveraging big data and prediction, 2016 27th Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC) (2016) 191–196.
[20] T. Hirai, M. Kano, Adaptive virtual metrology design for semiconductor dry etching process through locally weighted partial least squares, IEEE Trans. Semicond. Manuf. 28 (2) (2015) 137–144.
[21] E. Ragnoli, S. McLoone, S. Lynn, J. Ringwood, N. Macgearailt, Identifying key process characteristics and predicting etch rate from high-dimension datasets, Advanced Semiconductor Manufacturing Conference, 2009. ASMC'09. IEEE/SEMI, 2009, pp. 106–111.
[22] D. Zeng, C.J. Spanos, Virtual metrology modeling for plasma etch operations, IEEE Trans. Semicond. Manuf. 22 (4) (2009) 419–431.
[23] G.A. Susto, A. Schirru, S. Pampuri, S. McLoone, Supervised aggregative feature extraction for big data time series regression, IEEE Trans. Ind. Inform. 12 (3) (2016) 1243–1252.
[24] C. Park, S.B. Kim, Virtual metrology modeling of time-dependent spectroscopic signals by a fused lasso algorithm, J. Process Control 42 (2016) 51–58.
[25] R. Tibshirani, M. Saunders, S. Rosset, J. Zhu, K. Knight, Sparsity and smoothness via the fused lasso, J. R. Stat. Soc. Ser. B (Stat. Methodol.) 67 (1) (2005) 91–108.
[26] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[27] S. Lawrence, C.L. Giles, A.C. Tsoi, A.D. Back, Face recognition: a convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997) 98–113.
[28] T. Wang, D.J. Wu, A. Coates, A.Y. Ng, End-to-end text recognition with convolutional neural networks, 2012 21st International Conference on Pattern Recognition (ICPR), 2012, pp. 3304–3308.
[29] A. Romero, C. Gatta, G. Camps-Valls, Unsupervised deep feature extraction for remote sensing image classification, IEEE Trans. Geosci. Remote Sens. 54 (3) (2016) 1349–1362.
[30] W. Zhao, S. Du, Spectral-spatial feature extraction for hyperspectral image classification: a dimension reduction and deep learning approach, IEEE Trans. Geosci. Remote Sens. 54 (8) (2016) 4544–4554.
[31] Y. Sun, X. Wang, X. Tang, Deep learning face representation from predicting 10,000 classes, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2014) 1891–1898.
[32] F. Lin, K.-T. Cheng, An artificial neural network approach for screening test escapes, Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific, 2017, pp. 414–419.
[33] H. Lee, Y. Kim, C.O. Kim, A deep learning model for robust wafer fault monitoring with sensor measurement noise, IEEE Trans. Semicond. Manuf. 30 (1) (2017) 23–31.
[34] W. Yan, D. Tang, Y. Lin, A data-driven soft sensor modeling method based on deep learning and its application, IEEE Trans. Ind. Electron. 64 (5) (2017) 4237–4245.
[35] J. Choi, M.K. Jeong, Deep autoencoder with clipping fusion regularization on multistep process signals for virtual metrology, IEEE Sens. Lett. 3 (1) (2019) 1–4.
[36] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[37] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (December) (2010) 3371–3408.
[38] D.P. Kingma, M. Welling, Auto-Encoding Variational Bayes, 2013 arXiv preprint arXiv:1312.6114.
[39] D. Basak, S. Pal, D.C. Patranabis, Support vector regression, Neural Inf. Process.-Lett. Rev. 11 (10) (2007) 203–224.
[40] A.L. Maas, A.Y. Hannun, A.Y. Ng, Rectifier nonlinearities improve neural network acoustic models, Proc. ICML, vol. 30 (2013).
[41] D.E. Rumelhart, G.E. Hinton, R.J. Williams, et al., Learning representations by back-propagating errors, Cogn. Model. 5 (3) (1988) 1.
[42] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, L. Carin, Variational autoencoder for deep learning of images, labels and captions, Advances in Neural Information Processing Systems, 2016, pp. 2352–2360.
[43] K.J. Kanarik, T. Lill, E.A. Hudson, S. Sriraman, S. Tan, J. Marks, V. Vahedi, R.A. Gottscho, Overview of atomic layer etching in the semiconductor industry, J. Vac. Sci. Technol. A Vac. Surf. Films 33 (2) (2015) 020802.
[44] J.W. Coburn, H.F. Winters, Ion and electron assisted gas surface chemistry – an important effect in plasma etching, J. Appl. Phys. 50 (5) (1979) 3189–3196, doi:10.1063/1.326355.
[45] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on imagenet classification, Proceedings of the IEEE International Conference on Computer Vision (2015) 1026–1034.
[46] F. Chollet, Keras, 2015. http://github.com/fchollet/keras.
[47] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, 2016 arXiv preprint arXiv:1603.04467.
[48] R.R. Picard, R.D. Cook, Cross-validation of regression models, J. Am. Stat. Assoc. 79 (387) (1984) 575–583.
[49] A.A. Khan, J.R. Moyne, D.M. Tilbury, Virtual metrology and feedback control for semiconductor manufacturing processes using recursive partial least squares, J. Process Control 18 (10) (2008) 961–974.
[50] G.A. Susto, A. Schirru, S. Pampuri, G. De Nicolao, A. Beghi, An information-theory and virtual metrology-based approach to run-to-run semiconductor manufacturing control, 2012 IEEE International Conference on Automation Science and Engineering (CASE), 2012, pp. 358–363.
[51] C.-A. Kao, F.-T. Cheng, W.-M. Wu, F.-W. Kong, H.-H. Huang, Run-to-run control utilizing virtual metrology with reliance index, IEEE Trans. Semicond. Manuf. 26 (1) (2012) 69–81.
[52] G.A. Susto, A.B. Johnston, P.G. O'Hara, S. McLoone, Virtual metrology enabled early stage prediction for enhanced control of multi-stage fabrication processes, 2013 IEEE International Conference on Automation Science and Engineering (CASE), 2013, pp. 201–206.
[53] L. Puggini, S. McLoone, Forward selection component analysis: algorithms and applications, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2395–2408.