1D ResNet for Fault Detection and Classification on Sensor Data in Semiconductor Manufacturing

Philip Tchatchoua
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France
philip.tchatchoua@lis-lab.fr

Guillaume Graton
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Ecole Centrale Marseille
Marseille, France

Mustapha Ouladsine
Aix Marseille Univ, Université de Toulon, CNRS, LIS
Marseille, France

Julien Muller
EZAKO
Toulon, France

Abraham Traoré
EZAKO
Toulon, France

Michel Juge
STMicroelectronics
Rousset, France

2022 International Conference on Control, Automation and Diagnosis (ICCAD) | 978-1-6654-9794-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCAD55197.2022.9853997

***Abstract*—With much attention being placed on reducing manufacturing costs and improving productivity, maintaining process tools in good operating conditions is one of the most important objectives. A huge amount of data is collected during manufacturing processes and the challenge nowadays is to efficiently make use of this massive data. In this paper, a multivariate time-series fault detection method, based on the 1D ResNet algorithm is proposed. The objective is to analyze the raw data, collected via various sensors during the semiconductor manufacturing process in order to detect abnormal wafers. For this, a set of features derived from specific tools in the manufacturing chain are selected and evaluated to characterize the wafer status. Two distinct data sets are used to validate the proposed approach. The results obtained highlight the strengths of the proposed method, which could serve as a valuable decision-making support for abnormal wafer detection in the semiconductor manufacturing process.**

***Index Terms*—Fault detection, Raw multivariate sensor data, Deep learning, ResNet, Semiconductor manufacturing**

I. Introduction

Semiconductor manufacturing can be described as a batch multi-step process, involving sequences of photographic and chemical processing steps, numerous equipment types, with recurrent changes on operations, products, and processes. During equipment processing, products are grouped into batches of 25 silicon wafers. Each wafer goes through a sequence of operations during which electronic circuits are gradually crafted one layer after the other to create the final semiconductor product, obtained after extensive cycles (hundred operations) over several months. This complex and lengthy wafer manufacturing process is highly nonlinear, and subject to various disturbances. These disturbances can arise from equipment (aging, cleaning, repairs), product (wafer state) or process (preprocess chambers, warm-up). This leads to process variability from one wafer to another, within a batch, or between different batches. This can be observed via numerous sensors present on process equipment, used to automatically collect equipment data, which gives direct information on the process conditions such as temperatures, pressures, gas flows, flow rates, powers, capacitances, etc. This represents a massive amount of sensor data collected routinely and stored on appropriate media.

In order to ensure a continuous, consistent, and replicable production quality, the hundreds of variables collected from equipment are used for diverse purposes: equipment monitoring via fault detection and classification, diagnosis, prognosis [1], equipment health management, predictive maintenance [2], and virtual metrology [3]. The current paper focuses in particular on the usage of sensor data for fault detection and classification (FDC). Indeed, for each equipment, early detection and effective classification of faulty wafers resulting from abnormal processing is essential for controlling operations, reducing yield losses, and preventing abnormal wafers from reaching posterior stages.

Historically, FDC in the semiconductor industry is done by the univariate monitoring of sensor data variables, which are also referred to as status variable identification (SVID) [4]. For this, univariate indicators are crafted from summary statistics (usually mean and standard deviation but also maximum, median, range, variance, etc.), computed on user-defined time windows, for specific steps or whole recipes. A recipe is a sequence of elementary steps during which a specific operation is executed by an equipment unit for a given product. These univariate indicators are monitored using control charts. Even though this approach yields good results, it suffers from miscellaneous drawbacks such as the lack of consideration of static and dynamic relationships between multivariate SVIDs, partial process coverage, high sensitivity of the indicators to equipment events, a great number of control charts, etc. To circumvent these drawbacks, the research community introduced some more efficient FDC methodologies for the semiconductor industry.

A range of multivariate statistical process control methods have been studied, based on Principal Component Analysis (PCA) [5], [6]. These are self-supervised learning methods,

*Research supported by STMicroelectronics company and MadeIn4 European project under grant agreement No. 826589.

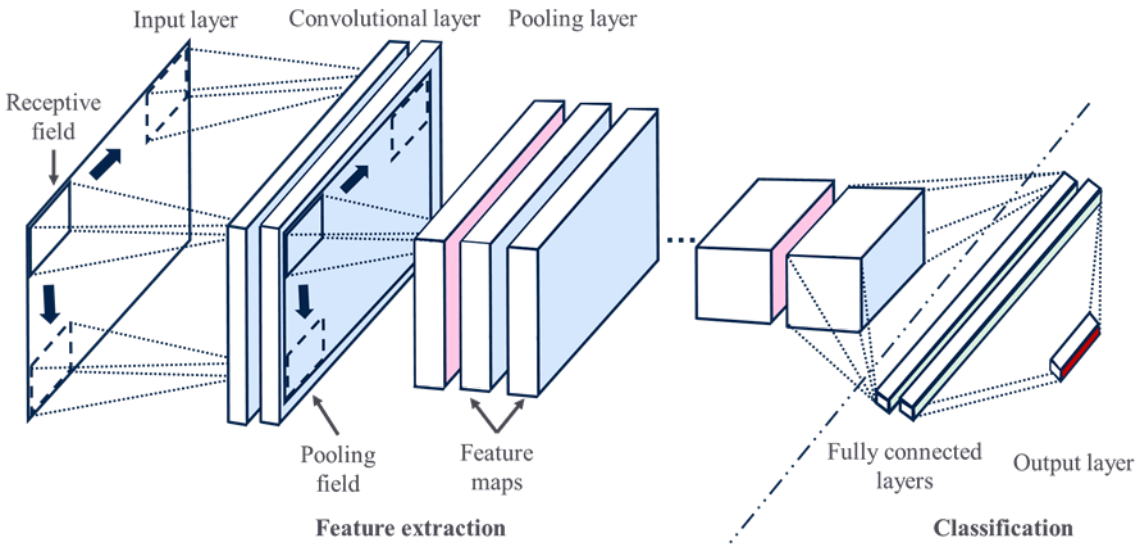


Fig. 1: Structure of a convolution neural network.

which are widely used for anomaly detection due to the scarcity of faulty wafers. Some hybrid methods combining the dimensionality reduction via PCA and supervised learning methods such as k-Nearest Neighbors (k-NN) [7] and Support Vector Machines (SVM) [8] have also been proposed for faulty wafer identification. With the paradigm of high data volumes and the complexity of sensor signals, these methods, needing extensive data preprocessing show some limitations.

In order to adequately respond to the new big data challenges of the semiconductor industry, deep learning methods have been investigated for FDC [9], [10]. From Convolutional Neural Networks [11], [12] to autoencoders [13] and Recurrent Neural Networks [14], these deep learning methods (very successful on several tasks such as image and video processing, sequential data processing like text or time series, feature extraction) have been successfully modified for various process control tasks.

In this paper, a ResNet architecture is proposed for FDC on raw multivariate time series, which captures both the temporal dynamics and correlations with analysis on two data sets to show the efficiency of this approach with respect to the state-of-the-art.

The remainder of this paper is as follows: Section II introduces representative deep learning methods. Section III describes the proposed model while Section IV presents the experimental set up and detection performances on real and simulated data from a semiconductor manufacturer. Finally, Section V concludes the paper and discuss of future studies.

II. Deep learning models

In this section, neural networks approaches used for the experimental analysis are briefly presented. The underlying motivation of this choice stems from the fact they represent the benchmark deep learning based methods for fault analysis [11], [13], [14], [16].

A. Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) represent one of the standard methods for image processing in deep learning [15]. A CNN usually consists of two parts, as in Fig. 1. A first part for feature extraction, composed of an input layer, a convolutional layer, a pooling layer and a second part for classification, composed of fully connected layers and the output layer.

The purpose of the convolutional layer is to extract features from the data through the application of a filter and an activation function in order to capture the underlying non-linearity in the data. The pooling layer recovers the output of the convolutional layer and reduces the size of the intermediary algebraic elements while keeping as much as possible the discriminating power of the network. The convolutional layers craft features while the pooling layers do feature selection. The fully connected layers, which together form a multi-layer perceptron, use a one-dimensional array obtained from the three-dimensional shaped output of the last pooling or convolutional layer for the classification task.

The receptive field is a square matrix of weights which establishes a connection between the input layer and the convolutional layer. Having a size smaller than the input data, the receptive field sweeps across its horizontal and vertical axis with a predetermined stride and performs convolutions. For an input x size $T \times K$, the output of a convolution operation with no padding, stored in a node is given by:

$$y_{ij} = \sigma\left(\sum_{m=1}^{F}\sum_{n=1}^{F} w_{m,n} x_{(m+iS),(n+jS)} + b\right), \quad \text{for } 0 \le i \le \frac{T-F}{S}, \text{and } 0 \le j \le \frac{K-F}{S}, \tag{1}$$

where F is the size of the square receptive field; S is the step size of the receptive field commonly called

stride; $x_{(m+iS)(n+jS)}$ is the input element at position $(m+iS, n+jS)$; $w_{m,n}$ and b are the weight at position (m, n) and the bias, respectively; σ is a non-linear activation function, usually a rectified linear unit (ReLU). The receptive field or filter applied to create a feature map has a single weight matrix, thus all the nodes of a feature map share the same weights. Due to this characteristic, the receptive field searches for a feature of a common characteristic (e.g., a single inter-variable correlation in the multivariate sensor signals) across the entire input data [11]. In any convolutional layer, the receptive field is applied iteratively with different weight matrices creating multiple feature maps with different features crafted in each of them. When computing the operation in (1), the size of the output feature maps is reduced to $\left(\frac{T-F}{S}+1\right) \times \left(\frac{K-F}{S}+1\right)$. Consequently, the output dimension gradually decreases as the network becomes deeper. For the output size to be maintained before pooling, a padding operation can be done.

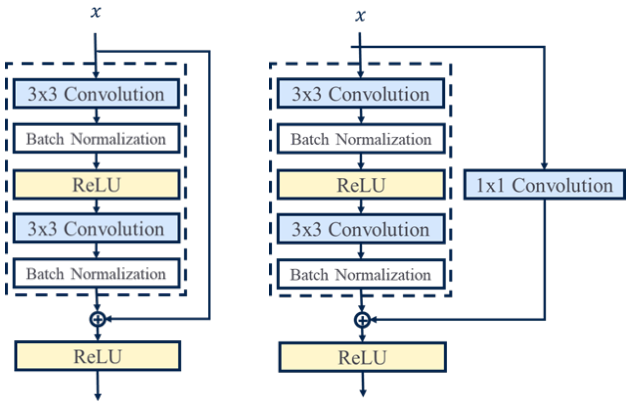
B. Long Short Term Memory (LSTM)

Long Short Term Memory, also called LSTM, is a special type of neural networks proposed to capture the temporal dynamics of a sequential data (e.g. time series). The architecture of LSTM is composed of units called memory blocks. A memory block contains cells with self-connections storing the temporal information of a given network in addition to some multiplicative gates to control the information flow. Each memory block contains an input gate controlling the input activation flow, an output gate controlling the output flow and a forget gate which scales the internal state of a cell and adaptively forgets or resets the memory of a given cell.

Recently, the so-called attention mechanism has drawn interest in the fault analysis community. Indeed, several architectures have been proposed [14], [16]. Attention is unarguably one of the most recent and efficient concepts in deep learning since it leads to breakthroughs in the field in terms of performance improvement [17]. Roughly speaking, the principle of the attention mechanism is to give a model the ability to focus only on a specific part of the data. For example, it allows an LSTM to focus on specific parts of a sentence or a time series. In [12] a self-attentive CNN method is proposed for fault detection, where the pooling layer is replaced by self-attention.

C. Residual Neural Networks (ResNet)

Residual Neural Networks (ResNet) are an improved version of Convolutional Neural Networks (CNN). They have been introduced in order to alleviate the optimization difficulty of Deep Convolutional Neural Networks [18]. With the increase in a network depth, its performance saturates, and eventually deteriorates, not due to over-fitting, but due to the vanishing gradient noticed in deep architectures [19]. Numerous deep network architectures such as Highway Networks [20], DenseNet [21], ResNet [18], [19] have been proposed to tackle this problem and have consistently shown state-of-the-art performances in different typologies. The underlying idea of ResNet is to skip one or more convolutional layers by using shortcut connections to form shortcut or residual blocks. It has been proven empirically that these blocks significantly improve the training while alleviating the degradation problem [19] and eases optimization. Also, deep networks with skip connections do not have a higher error than the shallow equivalent networks.



(a) Identity block (b) Convolution block

Fig. 2: Identity and convolutional block with skip connection.

On the hypothesis that multiple nonlinear layers can approximate complex functions and assuming that their input and output have the same dimensions, then these layers can approximate residual functions [18]. The shortcut connections make the learning of such functions much easier. ResNet adopt residual learning for every few stacked layers, in the form of building blocks. The shortcut connections can simply perform identity mapping as in the identity block in Fig. 2a or can perform a linear projection as in the convolution block in Fig. 2b. For the identity blocks, their output is added to the output of the stacked layers and this adds no extra parameter or computational complexity to the network. They thus have the same number of parameters, depth, width and can be easily compared to the corresponding plain networks. For an input x, their output y is defined as:

$$y = \sigma\left(\mathcal{F}\left(x, \{W_i\}\right) + x\right) \tag{2}$$

where $\mathcal{F}\left(x, \{W_i\}\right)$ is the residual mapping to be learned and σ the ReLU activation function. The function $\mathcal{F}\left(x, \{W_i\}\right)$ can represent multiple convolutional, normalization and activation layers, for which element-wise addition is performed on two feature maps, channel by channel. For the convolution blocks, the shortcut connections perform a linear projection in order to match dimensions between the input x and the residual mapping $\mathcal{F}\left(x, \{W_i\}\right)$ The output of this block is:

$$y = \sigma\left(\mathcal{F}\left(x, \{W_i\}\right) + W_s x\right) \tag{3}$$

where W_s is a square matrix, performing the linear projection of x. This is used when a change in dimension occurs in the stacked layers of the block. The form of the residual blocks is flexible. Those blocks presented in Fig. 2 have 2 convolutional layers but more layers are possible and diverse configurations also.

III. PROPOSED METHOD FOR FAULT DETECTION

For our fault detection task, a ResNet-type architecture is proposed. As stated earlier, ResNet is an improved version of the classical convolutional network which reduces the training difficulty of deep neural networks. The architecture consists of a succession of residual blocks, convolution, batch normalization, dropout, pooling layers for feature extraction as well as fully connected layers for classification.

The conventional square receptive field of CNNs is not tailored to extract inter-variable and temporal correlations among all the SVIDs, which is necessary for fault detection on multivariate time series data. The proposed architecture copes with this by adopting a rectangular receptive field, which moves along the time axis only. For an input wafer x of size $T \times K$, which represents T SVIDs and K time steps, the output of the first convolution operation with no padding, immediately after the input layer for a node is given by:

$$y_i = \sigma\left(\sum_{m=1}^{F}\sum_{n=1}^{T} w_{m,n} x_{(m+i\times S),(n)} + b\right), \quad 0 \le i \le \frac{K-F}{S}, \tag{4}$$

where F and S are the row size and the stride length of the receptive field, respectively. The architecture proposed in this paper, with identity blocks (Res-block a) and convolution blocks (Res-block b) is shown in Fig. 3.

The underlying motivation of the Batch Normalization layer is to reduce the computational complexity of the training. The dropout, which is performed through the spatial dropout layer, can be seen as a regularization of the network weights, leading to the mitigation of over-fitting. The residual blocks alleviates the degradation problem while extracting discriminating features from the data set. Here the two block types, identity and convolution are used. Two design rules are used for convolution layers of the blocks : (i) in one block, when the feature map size is the same, layers have the same number of filters ; (ii) when the feature map size is halved, the number of filters per layer is doubled. In the blocks, the halving or down-sampling is done by convolution layers with a stride of 2. The pooling layer reduces the dimension of the intermediary algebraic elements which is then flattened to set the 1D dimension needed by the fully connected layers.

IV. EXPERIMENT

The goal with respect to the numerical experiment is to prove that the proposed approach, based on ResNet, performs better than state-of-the-art neural networks approaches for fault analysis.

A. Data and setup

In order to demonstrate the effectiveness of the proposed model, two datasets provided by STMicroelectronics Rousset 8" fab are studied in this paper. To reduce the complexity, here, a focus is done on one equipment and one recipe for each dataset. The raw data used represents time series, consisting of 3-dimensional information, being wafer, variable and time. This raw data is nonetheless represented as a 2-dimensional matrix with processing time and SVIDs axis only. For both datasets, sensor data is collected every second.

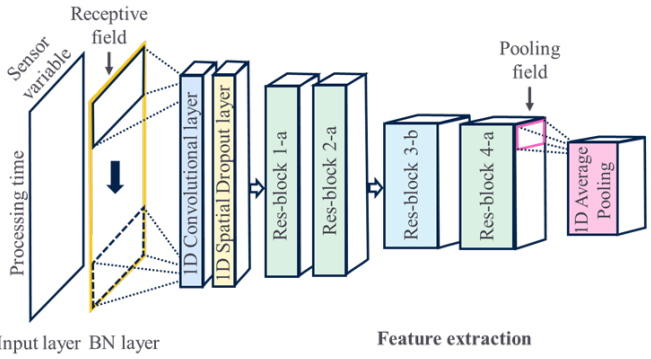


Fig. 3: Sequence of layers in the proposed model. BN and Res refer respectively to Batch Normalization and Residual.

The first dataset is from a simulation which mimics the dynamics of real variables like gas flows, pressures, temperatures of an etch tool. For a unique recipe lasting averagely 150 seconds, 11 variables are monitored for a total of 7000 wafers with 5000 normal samples and 2000 faulty samples This gives a ratio of 28.6% of faulty data, which is a good one with regards to the rarity of faulty data in the semiconductor industry.

The second dataset is from a plasma etching tool. The related production recipe consists in a series of 9 steps and lasts 130 seconds on average. Data is collected from numerous sensors present on the process tool. Among all SVIDs, domain engineers selected a set of 26 for fault detection and classification. For one month of production, this represents 516 wafers with 423 normal samples and 93 faulty samples. The ratio of faulty wafers is 18.0%, showing a class imbalance.

B. Neural networks configuration

For the comparison purpose, six neural networks models are considered: ResNet, SAE, CNN, CNN with self attention (AttentionCNN), LSTM, LSTM with self attention (AttentionLSTM). The AttentionCNN and AttentionLSTM correspond respectively to CNN and LSTM architectures with self attention layers. SAE corresponds to a stacked autoencoder, which is composed of two symmetrical artificial neural networks in a bottleneck form.

In terms of fine-tuning, several configurations have been evaluated and only the best parameters kept to generate the final results. The neural network architectures proposed for the experimental setting relies on the well-celebrated Tensorflow software.

The ResNet architecture is structured through: one convolutional layer (with 64 filters, a kernel size of 3 and a stride of 1), one spatial dropout layer (with a rate of 10%), four residual blocks (2 identity blocks with 64 filters followed by 1 convolution block and 1 identity block with 128 filters), one average pooling layer (the pool size being fixed to 2), one

dense layer (with 100 units), one dropout layer (with a rate of 10%). For the remaining layers, the parameters correspond to the default values proposed by Tensorflow.

The SAE is a fully connected layer-based model, defined by an encoder and a decoder network composed of dense layers with the decoder being the mirrored version of the encoder. The encoder has three hidden layers of 130, 90, and 60 nodes. The ReLU function is used as the activation function. The structure of the CNN is configured as follows: two convolutional layers with 20 and 40 filters coupled with 2 max-pooling layers (the pool size and the stride being respectively 2 and 3), one dense layer (with 100 units), one spatial dropout layer (rate: 10%) and one dropout layer (rate: 10%). For CNN with an attention mechanism, Luong-style attention is used. For the LSTM architecture, two layers with 30 LSTM cells each are used. Besides, one dropout layer (the rate being 10%) and one dense layer (with 100 units) are used. For the attention-based LSTM, the Luong-style attention is applied. In terms of activation function, the underlying non-linearity in the data is enforced through the sigmoid function.

C. Other configurations

- Data partitioning: the data set is split into training and test sets according to the ratio 80%-20%. This process is performed through a stratified 5-fold partitioning in order to avoid biased results. In terms of implementation, the partitioning is carried out via Scikit-learn.
- Weight initialization and optimization: the initial weights are defined by the Glorot uniform distribution. The Adaptive Moment Estimation (Adam) optimizer is used for the training. The learning rates for Adam are respectively fixed to 0.0005 for ResNet, SAE, LSTM, AttentionLSTM and 0.0001 for CNN and AttentionCNN. The batch sizes are respectively fixed to 32, 16, 64, 32, 16, 16 for ResNet, SAE, CNN, AttentionCNN, LSTM, AttentionLSTM. For all of the models, the number of epochs is fixed to 300 with early stopping and the cost-function is the binary cross entropy.

D. Evaluation metrics

The evaluation metrics are the F-scores for model efficiency assessment and the computational complexity. The F-score is a function of the *Precision* and the *Recall*. In this specific framework, the *Precision* (5) is the ratio of the actual faults among the total detected faults and the *Recall* (6) corresponds to the ratio of the actual faults with respect to the correct predictions. Given (5) and (6), the F-score is given by (7). We use as main score the $F_{weighted}$ as in (8) which is a weighted sum of F_0 and F_1 that takes into account the imbalanced data set. It follows:

$$Precision = \frac{T_P}{T_P + F_P}, \tag{5}$$

$$Recall = \frac{T_P}{T_P + F_N}, \tag{6}$$

$$F_\beta = \frac{(1+\beta^2) Precision.Recall}{\beta^2 Precision + Recall}, \tag{7}$$

$$F_{weighted} = \frac{F_1 A_P + F_0 A_N}{A_P + A_N}, \tag{8}$$

where T_P, F_P, F_N, A_P, A_N are true positive, false positive, false negative, actual positive and actual negative, respectively.

The efficiency of a given model is an increasing function of the score, *i.e.* the model is considered very precise when the score is high (close to 1, which is the maximum upper bound).

Remark: it is worth highlighting that accuracy, which is the most intuitive way to evaluate a model, is not a convenient efficiency measure for an imbalanced data set [22].

E. Results and discussion

Tables I and II present the results of the proposed method and the aforementioned deep learning based fault detection methods on the simulated and the real data set, respectively. They are the best results for the fault detection models, obtained with optimized hyperparameter values (best values for learning rate, batch size, number of epochs, dropout obtained by testing several configurations).

For both data sets in Tables I and II, the proposed model outperformed the other models by a significant margin and achieved the best performances with the highest $F_{weighted}$ scores (0.9424 and 0.9708). The LSTM based methods showed the worst scores and severely underperformed with regards to the other models. Both models scored NA for F_1 and $F_{weighted}$ due to the inability of the models to converge, and this on either data sets. The CNN based methods had the second best performance on the simulated data set and outperformed the fully connected based SAE model. Nonetheless, on the real data set, the SAE model had the second best performance with only slightly better $F_{weighted}$ score ($<$1%) than the CNN model. On the real data set, the simple CNN model does better than AttentionCNN model by a rather big margin ($>$10%), meanwhile the margin is very tight on the simulated data set ($<$1%). For all models, the F_0-score is always better than the F_1-score by some weighty margin which gives insight on the detection capacities of the models.

The F_0-score tells how well the models identify normal samples as normal while the F_1-score tells how well the

TABLE I: Scores for the simulated data set

Methods	F_0	F_1	$F_{weighted}$
ResNet	0.9624	0.8926	0.9424
SAE	0.9093	0.6796	0.8436
LSTM	0.8333	NA*	NA
AttentionLSTM	0.8333	NA	NA
CNN	0.9377	0.8059	0.9001
AttentionCNN	0.9324	0.7851	0.8903

*NA = Not Applicable

TABLE II: Scores for the real data set

Methods	F_0	F_1	$F_{weighted}$
ResNet	0.9825	0.9189	0.9708
SAE	0.9647	0.8421	0.9423
LSTM	0.8995	NA	NA
AttentionLSTM	0.8995	NA	NA
CNN	0.9659	0.8125	0.9379
AttentionCNN	0.9239	0.4167	0.8312

models identify faults as faults. From the above results, it can be said that all models globally identify normal samples quite well with F_0-scores all over 0.8 with the best scores at 0.9624 and 0.9825 obtained by the proposed model for the simulated and real data sets, respectively. Even though it is important to correctly identify normal samples, fault identification is the crucial point and an insight on this is given by the F_1-score. The LSTM based models have NA scores for F_1 due to their incapability to encode the lengthy sequences through time and are thus worthless on both used data sets.

Since the fault identification is more important than normal sample identification, the F_1-score provides a more revealing performance measurement. For all models, this score is lower than the F_0-score, with the best scores F_1-scores being 0.8926 and 0.9189 obtained by the proposed model for the simulated and real data sets, respectively. This implies that the models have more difficulties to identify faults than to identify normal samples and these difficulties vary from one model to another and can be quantified by the difference between both scores. This difference is rather big ($>10\%$) for all models except the proposed ResNet model, enlightening its betterment and positioning it as a reliable FDC method.

V. Conclusion

In this paper, a ResNet based fault detection method is proposed on multivariate sensor signals for semiconductor process monitoring. The proposed model redesigns the first convolutional layer in order to consider the structural characteristics of raw sensor data, and to enable the extraction of meaningful correlation and temporal information. In particular, the proposed method uses residual blocks with skip connections to improve the training while alleviating the degradation problem of deep neural networks. Experiments are conducted using simulated and real data from the semiconductor industry to set the performance of the proposed model. The proposed model outperforms the state-of-the-art and baseline models for fault detection. We here underline that the usage of such a small real data set for the training and testing can result in a lower degree of generalizability of the conclusions.

For future studies, data augmentation and transfer learning can be considered when training the model, to improve its generalization and detection performances. Also, future works can be conducted to adapt the model to be able to work on variable-length sensor data and give insights for fault diagnosis. Research will be required to not only correctly detect faults but also classify them following their nature and diagnose the equipment root causes.

Acknowledgment

We would like to thank one of our colleagues, Edoardo Salvini, for insightful exchanges on deep learning models.

References

[1] T. Wang, M. Qiao, M. Zhang, Y. Yang, and H. Snoussi, "Data-driven prognostic method based on self-supervised learning approaches for fault detection," Journal of Intell. Manuf., pp. 1611–1619, 2020. DOI:10.1007/s10845-018-1431-x

[2] P. Scheibelhofer, D. Gleispach, G. Hayderer, and E. Stadlober, "A methodology for predictive maintenance in semiconductor manufacturing", AJS, vol. 41, no. 3, pp. 161–173, 2016.

[3] R. Clain, V. Borodin, M. Juge, and A. Roussy, "Virtual metrology for semiconductor manufacturing: Focus on transfer learning," IEEE CASE, pp. 1621-1626, 2021.

[4] E. L. Park, J. Park, J. Yang, S. Cho, Y.-H. Lee, and H.-S. Park, "Data based segmentation and summarization for sensor data in semiconductor manufacturing," Expert Syst. Appl., vol. 41, no. 6, pp. 2619–2629, 2014. doi: 10.1016/j.eswa.2013.11.001

[5] A. Thieullen, M. Ouladsine, and J. Pinaton, "Application of PCA for efficient multivariate FDC of semiconductor manufacturing equipment," Advanced Semi. Manuf. Conf., 2013. DOI: 10.1109/ASMC.2013.6552755

[6] J. Marino, F. Rossi, M. Ouladsine, and J. Pinaton, "Gaussian Time Error: A new index for fault detection in semiconductor processes," Amer. Control Conf., 2016. DOI: 10.1109/ACC.2016.7525416

[7] Q. P. He, and J. Wang, "Principal Component based K-Nearest Neighbor Rule for Semiconductor Process Fault Detection," Proc. Amer. Control Conf., pp. 1606–1611, 2008.

[8] J. Park, I.-H. Kwon, S.-S. Kim, and J.-G. Baek, "Spline regression based feature extraction for semiconductor process fault detection using support vector machine," Expert Syst. Appl., vol. 38, no. 5, pp. 5711–5718, 2011. doi: 10.1016/j.eswa.2010.10.062.

[9] S.R. Saufi, Z.A.B. Ahmad, M.S. Leong, and M.H. Lim, "Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: a review," IEEE Access, vol. 7, pp. 122644–122662, 2019. DOI: 10.1109/ACCESS.2019.2938227

[10] P. Tchatchoua, G. Graton, M. Ouladsine, and M. Juge, "A comparative evaluation of deep learning anomaly detection techniques on semiconductor multivariate time series data," IEEE CASE, pp. 1613-1620, 2021.

[11] K. B. Lee, S. Cheon, and C. O. Kim, "A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes," IEEE Trans. Semi. Manuf., vol. 30, no. 2, pp. 135–142, 2017. DOI: 10.1109/TSM.2017.2676245.

[12] E. Kim , S. Cho , B. Lee, and M. Cho, "Fault detection and diagnosis using self-attentive convolutional neural networks for variable length sensor data in semiconductor manufacturing," IEEE Trans. Semi. Manuf., vol. 32, no. 3, 2019. DOI: 10.1109/TSM.2019.2917521

[13] H. Lee, Y. Kim, and C. O. Kim, "A deep learning model for robust wafer fault monitoring with sensor measurement noise," IEEE Trans. Semi. Manuf., vol. 30, no. 1, pp. 23–31, 2017. DOI: 10.1109/TSM.2016.2628865.

[14] Q. Zhang, J. Zhang, J. Zou, and S. Fan, "A novel fault diagnosis method based on stacked LSTM," IFAC-PapersOnLine, vol. 53, no. 2, 2020.

[15] M. Browne, and S. Ghidary, "Convolutional Neural Networks for Image Processing: An Application in Robot Vision," Aust. Conf. on Artificial Intell., pp. 641-652, 2003.

[16] L. Xiang, P. Wang, X. Yang, A. Hu, and H. Su, "Fault detection of wind turbine based on SCADA data analysis using CNN and LSTM with attention mechanism," Measurement, 2021. DOI: 10.1016/j.measurement.2021.109094.

[17] A. Vaswani et al., "Attention is all you need," Advances in Neural Information Processing Syst., pp. 5998–6008, 2017.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," IEEE Conf. on Computer Vision and Pattern Recog., pp. 770-778. 2016.

[19] A. Zaeemzadeh, N. Rahnavard, and M. Shah, "Norm-preservation: Why residual networks can become extremely deep?" IEEE Trans. on Pattern Analysis and Machine Intell., 2020.

[20] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," Inter. Conf. on Neural Information Processing Syst., 2377–2385, 2015.

[21] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q.Weinberger, "Densely connected convolutional networks," IEEE Conf. on Computer Vision and Pattern Recog., pp. 2261–2269, 2017.

[22] L.A. Jeni, J.F. Cohn, and F. De la Torre, "Facing imbalanced data recommendations for the use of performance metrics," Int. Conf. on Affective Computing and Intell. Interac., 2013. DOI: 10.1109/ACII.2013.47