Data fusion by artificial neural network for hybrid metrology development

Conference Paper · February 2021

DOI: 10.1117/12.2583590

Lucien Penlap
ASML
4 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Jérôme Reche
Atomic Energy and Alternative Energies Commission
22 PUBLICATIONS **23** CITATIONS
SEE PROFILE

Guido Rademaker
Atomic Energy and Alternative Energies Commission
26 PUBLICATIONS **35** CITATIONS
SEE PROFILE

Data fusion by artificial neural network for hybrid metrology development

Penlap Woguia, L., Reche, J., Besacier, M., Gergaud, P., Rademaker, G.

L. Penlap Woguia, J. Reche, M. Besacier, P. Gergaud, G. Rademaker, "Data fusion by artificial neural network for hybrid metrology development," Proc. SPIE 11611, Metrology, Inspection, and Process Control for Semiconductor Manufacturing XXXV, 116112L (22 February 2021); doi: 10.1117/12.2583590

Event: SPIE Advanced Lithography, 2021, Online Only

Data fusion by artificial neural network for hybrid metrology development

L. Penlap Woguia[a], J. Reche[b], M. Besacier[a], P. Gergaud[b], G. Rademaker[b]
[a]Univ. Grenoble Alpes, CNRS, CEA/LETI-Minatec, Grenoble INP*, LTM, F-38054 Grenoble-France
[b]Univ. Grenoble Alpes, CEA, LETI, Platforms Department, F-38000, Grenoble, France
*Institute of Engineering and Management Univ. Grenoble Alpes

ABSTRACT

Hybrid metrology is a promising approach to access to the critical dimensions of line gratings with precisions. The objective of this work is about using artificial intelligence (AI), mainly artificial neural network (ANN) to improve metrology at nanoscale characterization by hybridization of several techniques. Namely, optical critical dimension (OCD) or scatterometry, CD–Scanning electron microscopy (CDSEM), CD–Atomic force microscopy (CDAFM) and CD–Small angle x-rays scattering (CDSAXS). With virtual data of tabular–type generated by modelling, the ANN is able to predict the geometrical parameters compared to true measured values with high accuracies and detect irregularities in input data.

Keywords: Dimensional metrology, hybridization, artificial intelligence, Neural network

1. INTRODUCTION

The industry of semiconductors continues to evolve at a fast pace, proposing a new technology node around every two years. Each new technology node presents reduced feature sizes and stricter dimension control. As the features of devices continue to shrink, allowed tolerances for metrology errors must shrink as well, pushing the evolution of the metrology tools.

No individual metrology technique alone can answer the tight requirements of the industry today, not to mention in the next technology generations. Besides the limitations of the metrology methods, other constraints such as the amount of metrology data available for higher order analysis and the time required for generating such data are also relevant and affect the usage of metrology in production. For the production of advanced technology nodes, neither speed nor precision may be sacrificed, which calls for cleverer metrology approaches, such as Hybrid Metrology.

Hybrid Metrology is based on the concept of the data fusion [1] and consists of employing different techniques of metrology together in order to combine their strengths while mitigating their weaknesses [2]. This hybrid approach goal is to improve the measurements in such a way that the final data presents better characteristics than each method separately. In previous work [3] the usual metrology techniques are used. It means the CD-SEM for its fast and almost-non-destructive metrology, the AFM-3D for its accurate profile view of patterns and non-destructive characteristic and scatterometry (named OCD) for its precision, global and fast measurements. In this article another technique is added, the CDSAXS, particularly well adapted for nanoscale dimensions with high precision [4]. The targeted patterns are lines characterized with 3 parameters: CD, height and Side Wall Angle (SWA). This article is focused on the development of the method to address the data coming from these different metrology techniques. The deep neural network (DNN) model is used to predict the main parameters (H, CD & SWA). The different steps to build a robust DNN are presented here, with simulated data to train and validate the DNN design. This work is carried out in the EU MadeIn4 project, which aims at proving the Industry 4.0 manufacturing productivity improvements by developing highly productive and connected cyber physical systems (CPSs) which combines metrology data analysis and design with machine learning (ML) methods. CEA/LETI is the sample provider and most of the measurements will be done with their metrology tools.

In this article, the first and second parts present the methodology used to design a neural network and the training data associated. The third part deals with the prediction parameters and the model performances. Then a last part shows the optimization studies and the convergence tests.

Metrology, Inspection, and Process Control for Semiconductor Manufacturing XXXV,
edited by Ofer Adan, John C. Robinson, Proc. of SPIE Vol. 11611, 116112L
© 2021 SPIE · CCC code: 0277-786X/21/$21 · doi: 10.1117/12.2583590

2. METHODOLOGY

The DNN was developed in Python by using the Keras application programming interface API [5] on TensorFlow [6] framework and Scikit Learn [7] library. The methodology detailed below, allow to explain how was designed the DNN. In this study, we considered that the data given from the different metrology techniques are the dimensional results (ie CD, H & SWA). Then consider the behavior of each technique is not mandatory for this kind of approach. However, the measurement uncertainties given by each technique are considered to build the database.

2.1 Data modeling

We understand by advanced numerical approach, a way of performing virtual experimental analyses by modelling the data obtained by each of the above-mentioned metrology tools. The first stage of data modelling consists in obtaining the data file containing the results of all the virtual techniques. For each of the three parameters (H, CD SWA), the virtual data is generated based on true values. The true values, or targeted values, are those that could have been measured by a technique and numerically expanded within a bounded interval.

The intervals of H and CD are centered on the targeted values (e.g. H = 50 ± 10 nm & CD = 20 ± 10 nm). However, the one of SWA is non-centered, starting from an initial value up to 90, which is the maximum angle corresponding to a rectangular pattern. The distribution of true values follows a uniform law, and their values were defined within fixed intervals, such as they uniformly cover the set range. Figure 1 is an example of graph displaying the uniform distribution of the true H and CD parameters. It shows that the data cover all the space delimited by the intervals of the respective parameters, which is an important condition for the generation of virtual data.

The data of virtual techniques were generated by the following equation (1).

$$datavirtual = datatrue \pm datatrue \times rand() \times uncertainty_{technique} \quad (1)$$

rand() is a random generator of numbers following a normal distribution in order to take into account to the Gaussian behavior of the uncertainty. It serves to standardize the distribution of the virtual values of parameters. To be as close as possible to experiments, the uncertainties in determining the geometrical parameters with each virtual technique ($uncertainty_{technique}$) was used. Figure 2 shows, in 2D the Gaussian distribution of virtual values generated. Table 1 resumes the uncertainties of parameters with the intervals of the true values used for their generation.

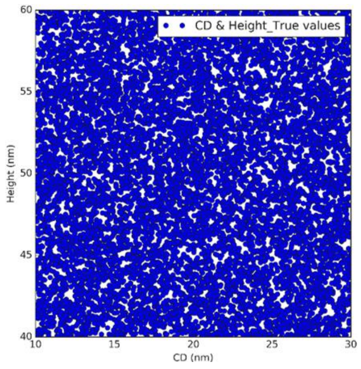


Figure 1. Uniform distribution of true values of height (H) vs CD.

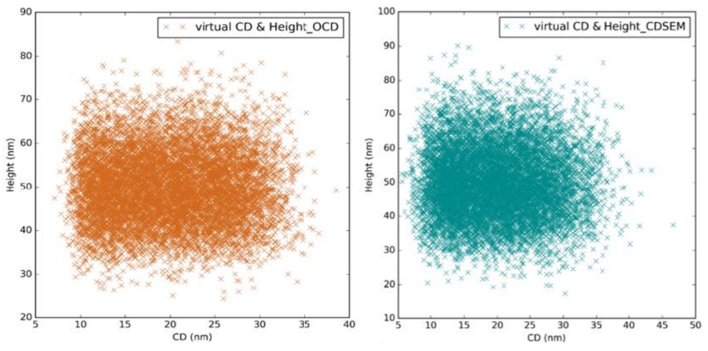
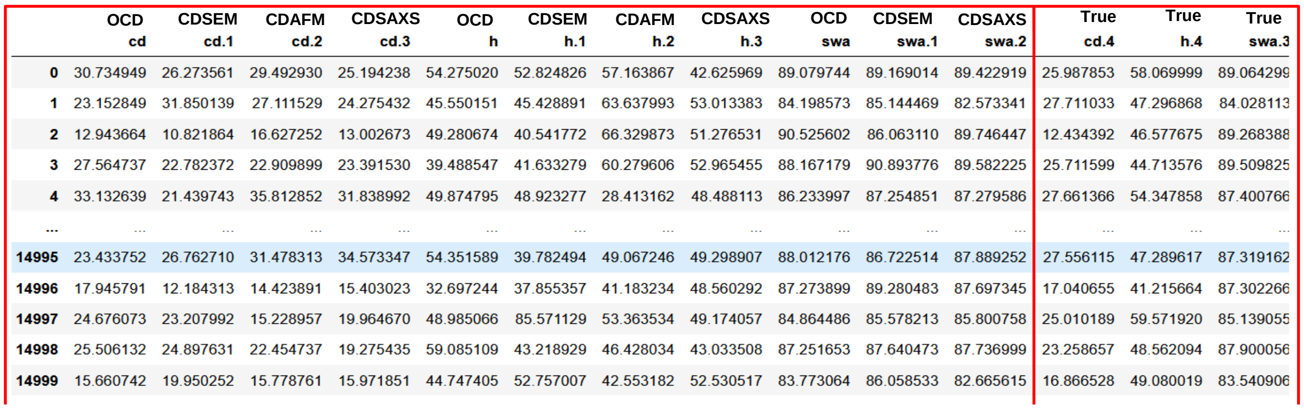


Figure 2. Gaussian distribution of virtual values of H vs CD. Example for 2 considered techniques

Parameters (True)	Height 40 < 50 < 60 nm	CD 10 < 20 < 30 nm	SWA 80° – 90°
Incertitude_OCD	12 %	10 %	1 %
Incertitude_CDSEM	17 %	15 %	1.2 %
Incertitude_CDAFM	30 %	22 %	
Incertitude_CDSAXS	15 %	10 %	1.1 %

Table 1. Uncertainties of different techniques used for data generation (for dimensional parameters)

Figure 3 illustrates the data file generated by using the virtual and true values. It was produced by using the Pandas library [8] in Python. The number columns of the entire data file are 14 (11 for virtual values and 3 for true values) and the number of rows is 15,000. This yields to a total size of 14 columns × 15,000 rows for the entire data file.



	OCD	CDSEM	CDAFM	CDSAXS	OCD	CDSEM	CDAFM	CDSAXS	OCD	CDSEM	CDSAXS	True	True	True
	cd	cd.1	cd.2	cd.3	h	h.1	h.2	h.3	swa	swa.1	swa.2	cd.4	h.4	swa.3
0	30.734949	26.273561	29.492930	25.194238	54.275020	52.824826	57.163867	42.625969	89.079744	89.169014	89.422919	25.987853	58.069999	89.064299
1	23.152849	31.850139	27.111529	24.275432	45.550151	45.428891	63.637993	53.013383	84.198573	85.144469	82.573341	27.711033	47.296868	84.028113
2	12.943664	10.821864	16.627252	13.002673	49.280674	40.541772	66.329873	51.276531	90.525602	86.063110	89.746447	12.434392	46.577675	89.268388
3	27.564737	22.782372	22.909899	23.391530	39.488547	41.633279	60.279606	52.965455	88.167179	90.893776	89.582225	25.711599	44.713576	89.509825
4	33.132639	21.439743	35.812852	31.838992	49.874795	48.923277	28.413162	48.488113	86.233997	87.254851	87.279586	27.661366	54.347858	87.400766
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
14995	23.433752	26.762710	31.478313	34.573347	54.351589	39.782494	49.067246	49.298907	88.012176	86.722514	87.889252	27.556115	47.289617	87.319162
14996	17.945791	12.184313	14.423891	15.403023	32.697244	37.855357	41.183234	48.560292	87.273899	89.280483	87.697345	17.040655	41.215664	87.302266
14997	24.676073	23.207992	15.228957	19.964670	48.985066	85.571129	53.363534	49.174057	84.864486	85.578213	85.800758	25.010189	59.571920	85.139055
14998	25.506132	24.897631	22.454737	19.275435	59.085109	43.218929	46.428034	43.033508	87.251653	87.640473	87.736999	23.258657	48.562094	87.900056
14999	15.660742	19.950252	15.778761	15.971851	44.747405	52.757007	42.553182	52.530517	83.773064	86.058533	82.665615	16.866528	49.080019	83.540906

Figure 3. Example of data file. The virtual values of techniques are in left part and the true values are in right part.

2.2 Data normalization

Within the framework of the neural network, it is advisable to normalize the input data. This is not mandatory but, if the data have different amplitudes, the weights, on the different connections in the network, will also have to be very varied, which would quickly unbalance the network and may lead to large errors in the predictions of parameters.
Several normalization methods are available in the literature, each of them having its advantages and limits according to the problematic and data types. In this work, we used the Inter Quartile Range (IQR) (Figure 4) [9]. It is known as a stable method to reduce outliers in the predictions.

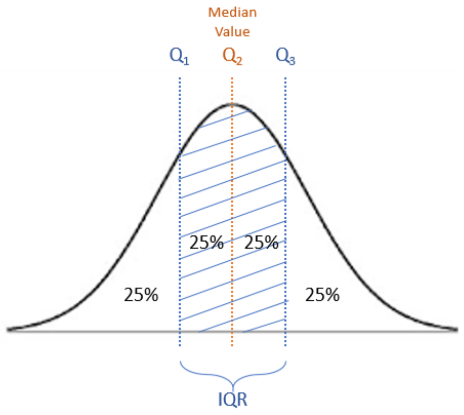


Figure 4. Graph of IQR

The formula used to normalize the data is given in equation (2).

$$Data_{Stand} = \frac{data - median}{IQR} \tag{2}$$

The IQR is the difference between the third and the first quartile (IQR = Q3 - Q1). The method consists in removing the median, then re-scale the data according to the quartile range. However the predictions of parameters, to compare with the expected ones (true values), must be done with the true features of data by using the DNN. Therefore, the inverse transform must be done.

3. DESIGN OF THE DEEP NEURAL NETWORK MODEL

In AI, a neuron is considered as a mathematical unit. A neural network (NN) is therefore obtained by connecting several neurons together [10]. A bare model of an artificial NN commonly known as perceptron is presented in Figure 5. The layers inserted between the input and the output layers are called hidden layers. The different layers in the NN have several neurons. Channels interconnect the layers. Each neuron are connected to each neuron of previous layer without exception. A neural network of such architecture is named as multiple layer perceptron (MLP) [11]. The information transmitted to the neurons of each layer via the channels are pondered by weights w_i. The decision of whether a neuron within a layer is activated or not, is made by an activation function (δ) that can be strictly linear or not. The output information consists of a linear combination of the weighted sum of inputs and bias. The right number of layer and neuron per layer have to be found to minimize errors during the processing.

The number of input neurons is known since they are determined by the structure of the input dataset. The number of neurons in the output layer are also determined according to the dimension of the expected information. Therefore, we consider them as the initially known hyperparameters. Concerning hidden layers and their number of neurons, there is no rule for their determination. That is why the commonly used method consists in trying several configurations until finding the best compromise for the final model architecture. An example of architecture is given Figure 6.

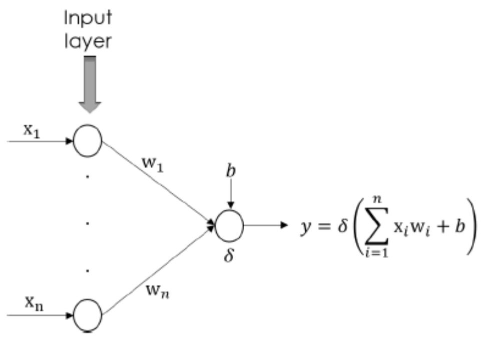


Figure 5. Single layer perceptron

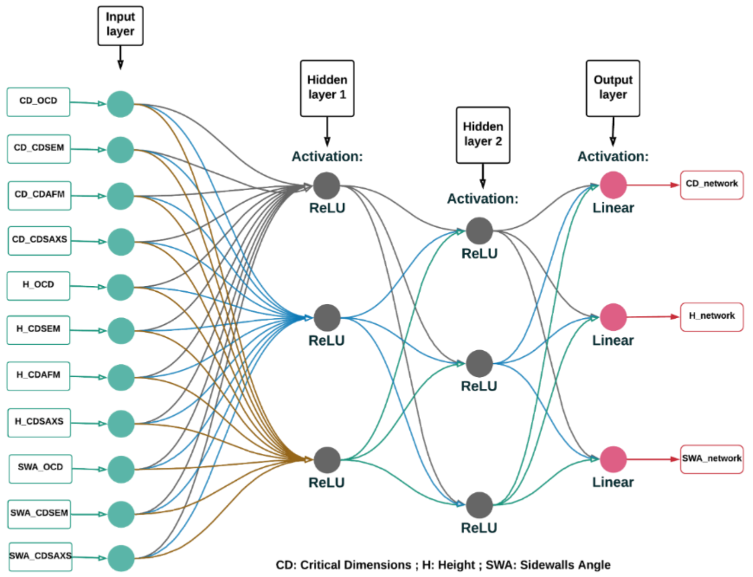


Figure 6. A multiple layer perceptron DNN model used for the simulations of virtual measurements and predictions of geometrical parameters of the patterns

3.1 Parameters and hyperparameters

We distinguish the parameters to the hyperparameters. A parameter is a data that vary and that is adjusted during the learning step. Examples of model parameters are the weights, the regression coefficients and support vectors in a SVM (support vector machine) [12]. However, a hyperparameter is an entity that is set before starting the training procedure. It is not intrinsic to the system but external and cannot be determined from the learning data. Hyperparameters are specified by the designer and can be used to help to optimize other models parameters. Some examples of hyperparameters are for instance: input, output and hidden layers, number of neurons, activation functions, batch size, number of epochs (iterations), learning rate, loss functions, optimizer…

- **Activation functions**: there are several types of activation functions dedicated to different purposes. Not all the activation functions can be used in any layers or for the same problematics. Functions such as tangent hyperbolic (Tanh), rectified Linear Unit (ReLU) and Leaky ReLU are generally used in hidden layers. Sigmod and linear activation functions are mostly used in the output layers. The activation functions used in these works are ReLU for hidden layers. It is well adapted for a quick learning. Since the problematic addressed here is of regression type, one of the best activations to use in output layer is a linear function. Neither activation function isassigned to the neurons of the input layer.

- **Loss function**: It computes the final error of the NN. The loss function used in these investigations to train models is binary-crossentropy
- **Batch size**: before describing batch, we introduce the concept of stochastic gradient descent. An interactive learning algorithm uses a training dataset to update the model. The Batch size is a hyperparameter of gradient descent that represents the number of samples to calculate the error used for back propagation per gradient updated. If batch size is not specified before the training phase, the default size considered will be 32.
- **Epoch**: it is also a hyperparameter of the gradient descend controlling the numbers of complete iterations through the training datasets.
- **Learning rate**: It is one of the most important parameters to accurately initialize when designing a NN model. The learning rate controls how to change the model in response to the estimated errors when the models weights are changed. It is used for the training of NN with values between 0 & 1. Smaller values of learning rates (< 0.5) require larger epochs and it is preferable to use few epochs when the learning rate is greater (>0.5).
- **Optimizer**: it is used for the search through different weights and any optional metrics of the network. The metric we used in the codes was sgd from Keras. This is an algorithm that automatically tunes itself and provides good results in wide range of problems.

3.2 Training & validation steps

The network is trained with a fraction of the overall data and the rest is dedicated to validation step. The ratio between data for training vs data for validation is chosen arbitrary but the number of data for training must be higher than those for validation. In our studies, this ratio has been adjusted. The starting point is for the training step 80 % of the overall data file and 20 % for validation step. This configuration for the choice of the training and validation data sizes was adopted based on research works reported in literature [13]. The training test split() module of the Scikit-Learn library was used, with a test size = 1 learning rate = 0.2.

4. PREDICTIONS AND PERFORMANCES

4.1 Parameters of prediction

The goal of the predictions performed by the neural network is to compare the outputs with the true values provided in the data file. An example of prediction study is shown in Figure 7. For the validation step, the data predicted by the DNN are compared to the true values.

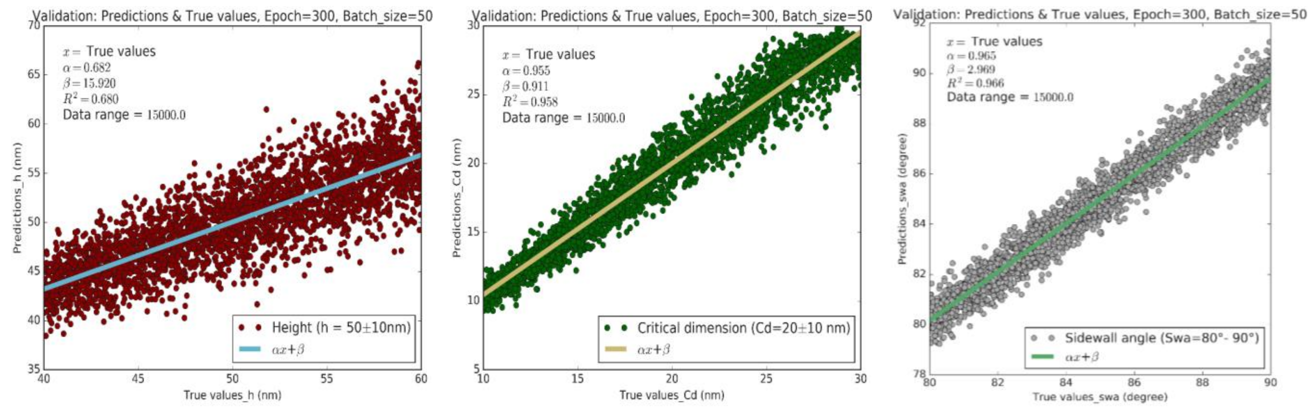


Figure 7. Comparison of the data predicted by the DNN model vs the true values. The solid lines represent the fitting of data by using a linear approximation. The coefficients of the line and the fit parameters are written on top left of the figures.

In these graphs, we observe a good fitting and the linear approximation is validated. The uncertainties that were assigned to each of the virtual techniques are the largest for the generation of H values and the lowest for SWA (see Table 1). It means that it could be better with a lower parameter uncertainties.

The quality of predictions is assessed by computing the mean absolute percentage errors (MAPE) (Equation 3).

$$MAPE = mean\left(\left|\frac{Y-\hat{Y}}{Y}\right|\right) \tag{3}$$

Y is the variable for expected (true) values and $\hat{Y}$ the variable for predicted values. The lower the MAPE values the better the capability of the model to predict the expected parameters with good accuracy. The MAPE computed after the predictions of the height, critical dimensions and sidewalls angles are 5.24 %, 4.74 % and 0.49 %, respectively (in example Figure 7). These values are low; it means a good performance of the model in the prediction of H, CD and SWA.

4.2 Model performances

The goal of this section is to analyze what happens when the initial dataset changes in the model of NN. We investigated the response of the model in terms of output errors computation. The main concepts that we focused on were the data structure and their uncertainties. The model of NN with 2 hidden layers and 3 neurons per layer (as shown in Figure 6) is used for this study.

Influence of data structures on the prediction errors

We focused on parameters CD and H because their intervals offer wide ranges of variation versus their true values, compared to SWA. Different cases were considered. With the centered value of H = 50 nm, the intervals explored were: 40 < H < 60 nm, 35 < H < 65 nm and 30 < H < 70 nm, respectively. Then, with the centered value of CD = 20 nm, the intervals chosen were also 10 < CD < 30 nm, 5 < CD < 35 nm, and 0 < CD < 40 nm, respectively. The results are presented in Figure 8 and Table 2. Figure 8 illustrates the impacts of different intervals on the predicted data. One observes a small increase of the prediction errors when the intervals widen for each parameter. Table 2 shows that the output errors, computed with the MAPE, tend to inscrease when the range of the input data increase. The case where 0 < CD < 40 nm is out of range because considering a zero CD does not have any signification.

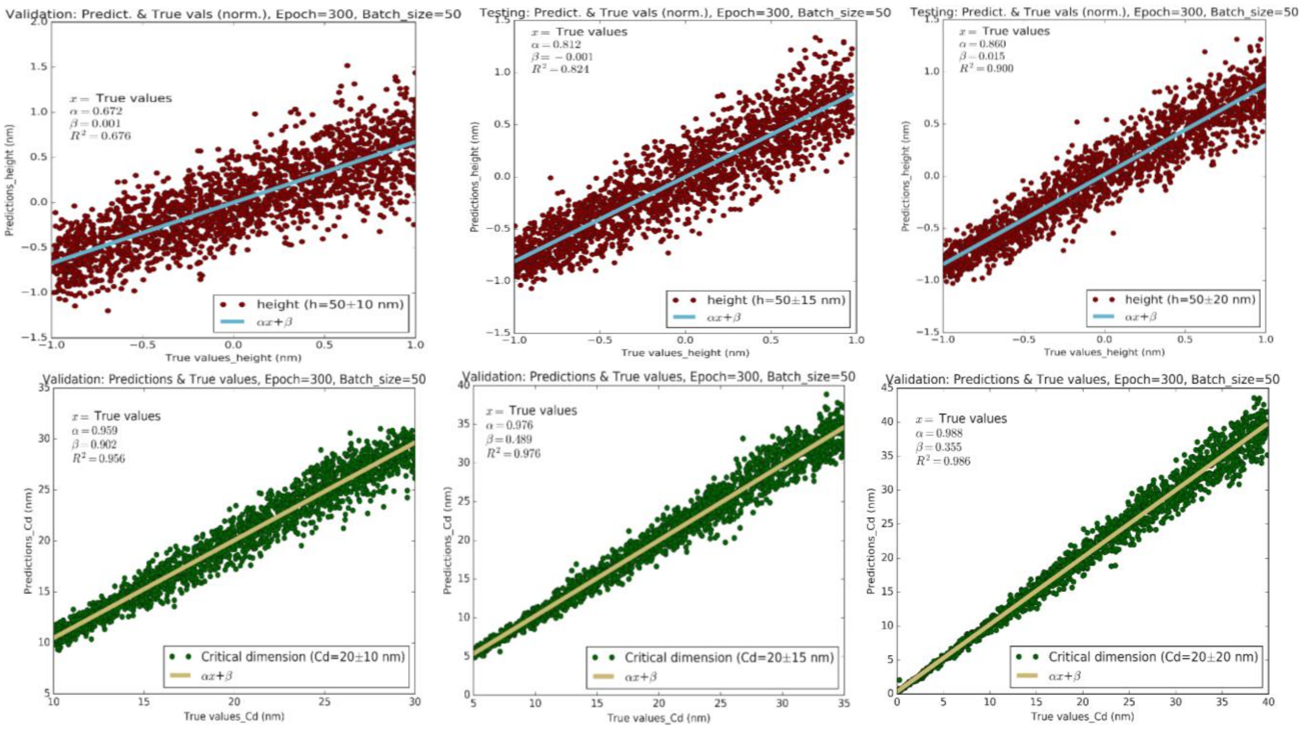


Figure 8. Comparison of the predicted values of H, Cd SWA by the network. The graphs were plotted with unstandardized data.

Height (nm)	Intervals:	$40 \leq 50 \leq 60$	$35 \leq 50 \leq 65$	$30 \leq 50 \leq 70$
	Errors: training set	5.31 %	5.88 %	6.07 %
	Errors: validation set	5.3 %	5.89 %	5.9 %
	Slope (α)	0.672	0.812	0.860
	R^2	0.676	0.824	0.9
CD (nm)	Intervals:	$10 \leq 20 \leq 30$	$5 \leq 20 \leq 35$	~~$0 \leq 20 \leq 40$~~
	Errors: training set	4.82 %	4.98 %	~~10.68 %~~
	Errors: validation set	4.81 %	5.1 %	~~14.68 %~~
	Slope (α)	0.959	0.976	~~0.988~~
	R^2	0.956	0.976	~~0.986~~

Table 2. Mean absolute percentage errors (MAPE) computed after the prediction of H and Cd in different intervals.

Influence of input errors

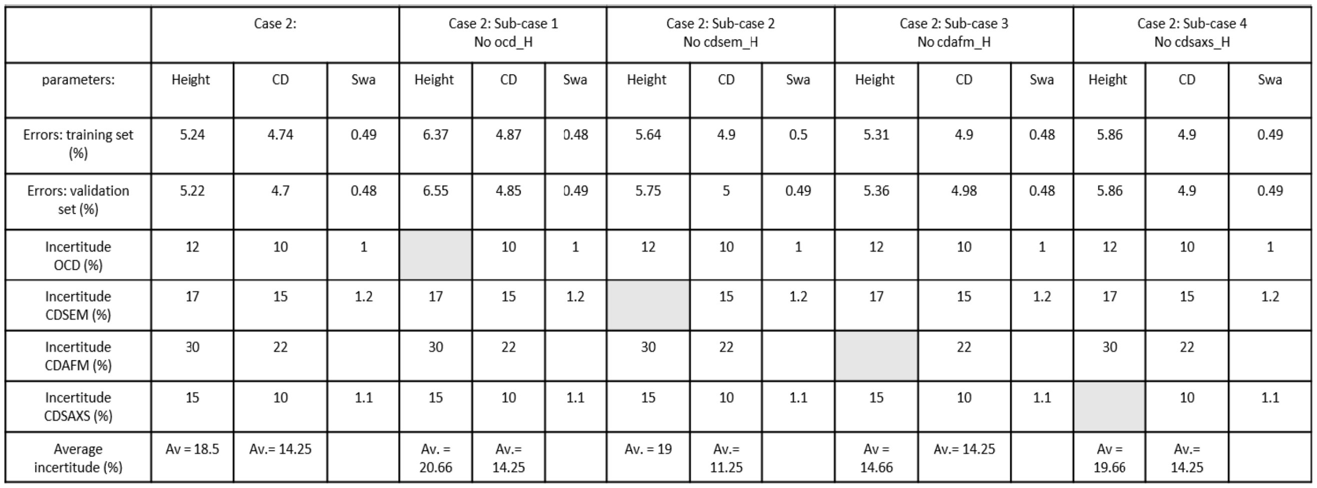
Here the goal is to check the sensitivity of the network to uncertainties in the input data and highlights their impact on the output results. To perform this, two cases are designed with different subcases as described in Table 3 and Table 4.

The Case 1 considered that the average values of uncertainties on H and CD are constant. Three uncertainties out of four were set with small and same values (6 %) but the fourth one was very large (26 %). In the subcase 1 the large uncertainty on CD was removed, in the subcase 2 this is the large uncertainty on height that was removed and in the subcase 3 both large uncertainties on CD and height was removed. Note that the uncertainties on SWA remained the same in the subcases. For subcase 1 the results on training and validation errors show a small decrease (a few tenths of a percent) on values for the CD but also for the height. The errors concerning the SWA increase. The trend is the same for subcase 2 and subcase 3 with the same order of magnitude for the training and validation errors. Then we can show that with or without the largest uncertainty on a given technique for one or two parameters, the NN stay stable and the impact on errors on training and validation is low (in a few tenths of a percent). It means that the NN is robust and can "absorb" the weakness of one technique in case on large measurement uncertainty.

	Case 1:			Case 1: Sub-case 1			Case 1: Sub-case 2			Case 1: Sub-case 3		
parameters:	Height	CD	Swa	Height	CD	Swa	Height	CD	Swa	Height	CD	Swa
Errors: training set	3.1 %	2.85 %	0.53 %	2.64 %	2.7 %	1 %	2.62 %	2.75 %	1 %	2.6 %	2.75 %	0.48 %
Errors: validation set	3.14 %	2.92 %	0.53 %	2.66 %	2.75 %	1 %	2.67 %	2.82 %	1 %	2.62 %	2.75 %	0.49 %
Incertitude_ OCD	6 %	6 %	1 %	6 %	6 %	1 %	6 %	6 %	1 %	6 %	6 %	1 %
Incertitude_ CDSEM	6 %	6 %	1.2 %	6 %	6 %	1.2 %	6 %	6 %	1.2 %	6 %	6 %	1.2 %
Incertitude_ CDAFM	26 %	6 %		26 %	6 %			6 %			6 %	
Incertitude_ CDSAXS	6 %	26 %	1.1 %	6 %		1.1 %	6 %	26 %	1.1 %	6 %		1.1 %
Cte. average incertitude (%)	Av.=11	Av.= 11		Av. = 11	Av.= 6		Av. = 6	Av. = 11		Av. = 6	Av. = 6	

Table 3. Study of training and validation errors on CD, H and SWA for different cases of measurement uncertainties on techniques. The initial case shows small values (6%) for 3 out of 4 techniques and a large value for the last technique.

Case 2 looks more realistic with the same uncertainties as those in Table 1. The average of uncertainties are different from each other and in the different subcases, one uncertainty out of four is removed for H parameter which vary (increase or decrease) the average value of uncertainty for H parameter. The results displayed in table 4 show that, as in the previous study, the removal of one uncertainty has a limited impact on the errors on the three parameters. The lowest errors are obtained when the average is low and the highest errors are obtained when the average is high. Nevertheless, this study also shows the robustness of the DNN.



	Case 2:			Case 2: Sub-case 1 No ocd_H			Case 2: Sub-case 2 No cdsem_H			Case 2: Sub-case 3 No cdafm_H			Case 2: Sub-case 4 No cdsaxs_H		
parameters:	Height	CD	Swa	Height	CD	Swa	Height	CD	Swa	Height	CD	Swa	Height	CD	Swa
Errors: training set (%)	5.24	4.74	0.49	6.37	4.87	0.48	5.64	4.9	0.5	5.31	4.9	0.48	5.86	4.9	0.49
Errors: validation set (%)	5.22	4.7	0.48	6.55	4.85	0.49	5.75	5	0.49	5.36	4.98	0.48	5.86	4.9	0.49
Incertitude OCD (%)	12	10	1		10	1	12	10	1	12	10	1	12	10	1
Incertitude CDSEM (%)	17	15	1.2	17	15	1.2		15	1.2	17	15	1.2	17	15	1.2
Incertitude CDAFM (%)	30	22		30	22		30	22			22		30	22	
Incertitude CDSAXS (%)	15	10	1.1	15	10	1.1	15	10	1.1	15	10	1.1		10	1.1
Average incertitude (%)	Av = 18.5	Av.= 14.25		Av. = 20.66	Av.= 14.25		Av. = 19	Av.= 11.25		Av = 14.66	Av.= 14.25		Av = 19.66	Av.= 14.25	

Table 4. Study of training and validation errors on CD, H and SWA for different cases of measurement uncertainties on techniques. The initial case shows values chosen in table 1. For each subcase, one uncertainty out of four is removed on H parameter.

Influence of data size

In the previous part, we have seen that the DNN model is able to minimize the influence of large input uncertainty (if any) to keep good performances level. These were observed for a defined data size. Here, the results are obtained by changing the size of dataset. We kept the data structure presented in Figure 6 and the uncertainties used in Table 1. Three data size were therefore chosen: 5000 rows, 10000 rows and 15000 rows. The plotting of the predicted data versus the expected ones are shown in Figure 9 and the MAPE in table 5. The errors computed for the three data sizes did not display remarkable variations. They remain constant though. Nevertheless, it does not forbid from proceeding to a more thoroughly analysis to check the upper and lower size limits at which the estimated output errors stabilize or not. This can be achieved by carrying out the convergence limits of the DNN model with different data sizes as presented in the next part.

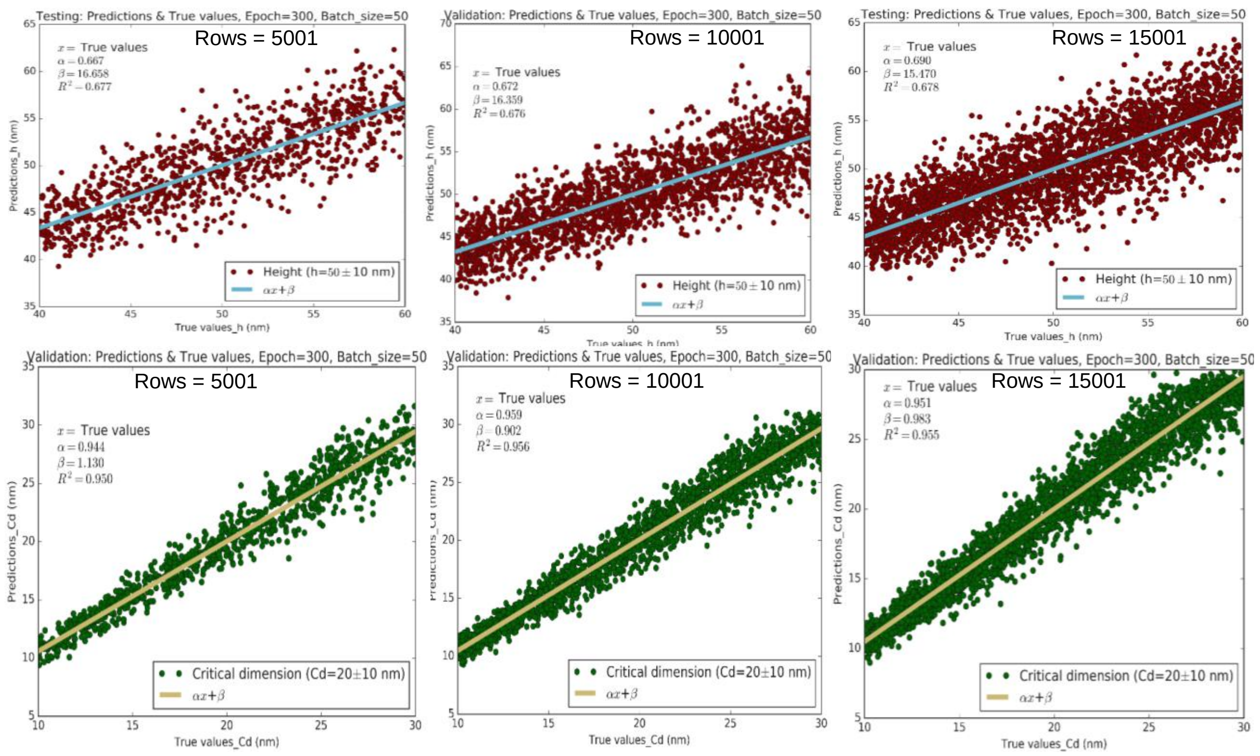


Figure 9. Comparison of the predicted values of H and CD by the DNN model with the true values, for different data sizes. The validation ratio was 20 % of the overall data set.

	Data range:	5001	10001	15001
Height 40<50<60 (nm)	MAPE: training set	5.24 %	5.26 %	5.21 %
	MAPE: validation set	5.22 %	5.25 %	5.20 %
	Slope (α)	0.667	0.672	0.69
	R^2	0.677	0.676	0.678
CD 10<20<30 (nm)	MAPE: training set	4.89 %	4.82 %	4.74 %
	MAPE: validation set	4.9 %	4.81 %	4.83 %
	Slope (α)	0.944	0.959	0.951
	R^2	0.95	0.956	0.955

Table 5. Presentation of the mean absolute percentage errors (MAPE) computed after the prediction of the parameters (H & CD) with different data sizes.

5. CONVERGENCE & OPTIMIZATION

5.1 Convergence tests

The convergence tests consist in training the DNN model with various data sizes, and then compute the training and validation losses. The size of the overall data file was 14 columns & 15000 rows. A fraction of rows varying from 2.5 % to 100 % was considered for both the training and validation stages. The number rows varies from 15000 x 2.5 % = 180 to 15000 x 100 % = 15000 rows. The ratios of 80% for the training and 20 % for the validation stages were set for each test.

Figure 10 shows the curves of convergence (loss function vs epoch) for all the data file sizes. It shows that for sizes below 1500 rows the convergence is not reached for number of epochs lower than 300. For sizes higher than 3000, the curves converge to the minimum of loss and require less epoch if the data size is high. For very high sizes of data file, we observed that the number of epoch does not vary so much. It means that there is a balance to find between the size of data file and the number of epoch required. In the example of the Figure 10, a data file of around 4000 rows seems to be a good compromise to converge with a limited number of rows.

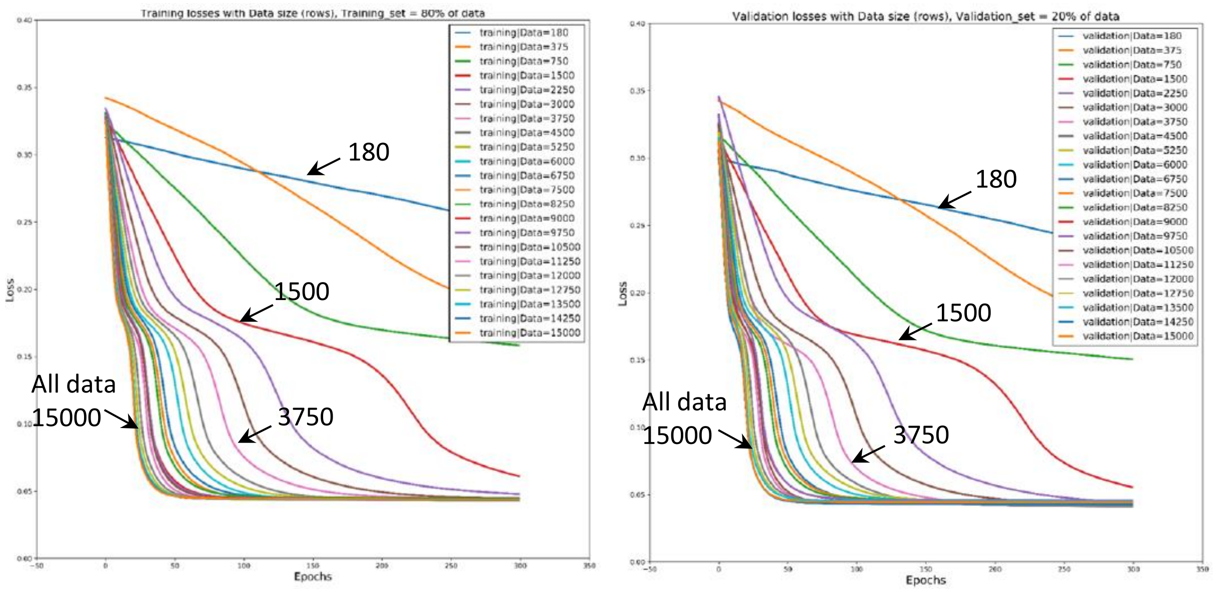


Figure 10. Study of the model convergence with deferent data size by assessing the evolution of the losses with the number of rows varying from 180 to 15000. The training and validation sizes were fixed respectively to 80 % and 20 % of data size. The total number of columns (=14) remains the same for all the cases.

We also computed the predictions of convergence for the three parameters for each data size. The results in Figure 11 show the computations of mean absolute percentage errors (MAPE) for the three parameters plotted versus data size divided in multiple of 15,000 rows. The number of epochs used here is 300. We observed that the three parameters converge rapidly,. The late convergence of CD indicates that this parameter imposes the convergence limits.

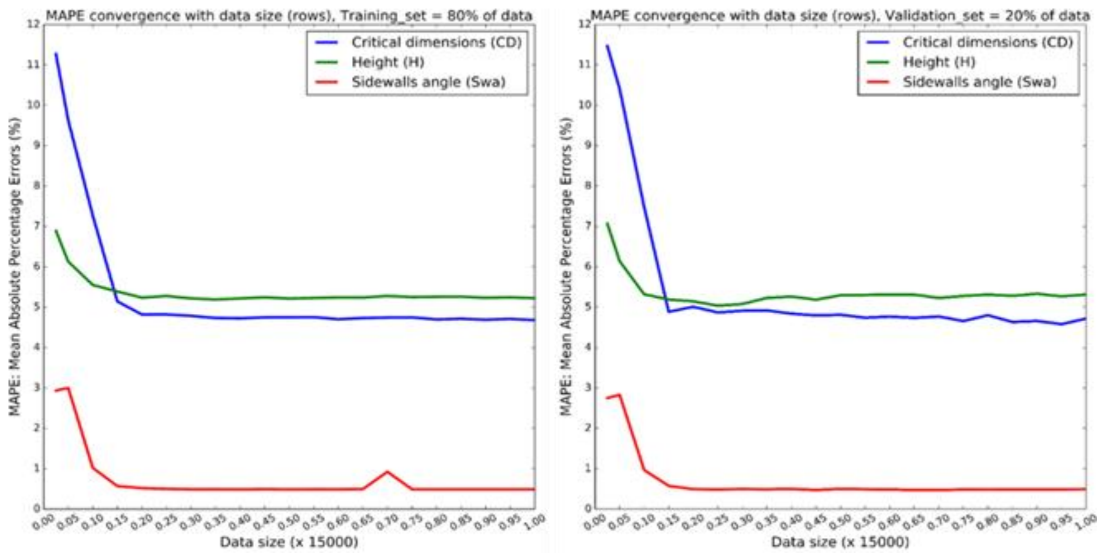


Figure 11. Evolution of MAPE for the three parameters vs data size in multiple of 15000 rows.

5.2 Optimization of models architecture

In this article, the process of model optimization consists in automatically training several models by varying the parameters and hyperparameters. A large number of models was tested. Each of them was trained 100 times and the convergence is reached if the conditions no MAPE for each parameter is reached (Equation 5). The goal is to know the complexity of the DNN model required to guarantee a convergence of the results. To implement these studies, the data were of size 15,000 rows & 14 columns; training size = 80% and validation size = 20%.

$$Training\ and\ Validation\ Phases \begin{cases} error_{CD} < 5\% \\ error_{H} < 5.6\% \\ error_{SWA} < 0.5\% \end{cases} \qquad (5)$$

More than 10 different DNN models were tested. The activation functions ReLU (in the hidden layers) & Linear (in output layer) were those with the best ratio of convergence results and was selected for the final tests. The errors computed with other activation functions (Tanh in hidden layers and sigmoid in output layers) were two fold above the conditions of equation (5) or more. For example, the non-convergences with Tanh & Sigmoid activation functions were observed because the error range were in the order of 22 % (CD), 11 % (H) and 1.7 % (SWA) in average. It means that these functions are not appropriate for our regression problem.
The results are resumed in the Table 6. 3 models among the 10 designs are presented. They was tested for different data sizes and different number of epoch. The configuration of models with X/Y1/Y2/Z means that it has X neurons in input layers, 2 hidden layers with Y1 and Y2 neurons per layers and finally Z neurons in output layers. In this study Y1 and Y2 are always equals. It shows that concerning the models presented here, the number of convergence is null or very low for a data size of 7,500 whatever the number of epoch or the number of neurons per hidden layer. For a data size of 15,000 the model 3 reaches the highest number of convergence with a high number of epoch and number of neurons per hidden layers. Then it means that the model must not be too simple to guarantee a good convergence level.

Activation = ReLU (hidden layers) and Linear (output layer) ; number of executions = 100						
Data size = 15000				Data size = 7500		
Epochs = 200				Epochs = 200		
Model	Model 1 : 11/3/3/3	Model 2 : 11/4/4/3	Model 3 : 11/5/5/3	Model 1 : 11/3/3/3	Model 2 : 11/4/4/3	Model 3 : 11/5/5/3
Nber. of convergence	30/100	75/100	72/100	None	None	None
Epochs = 300				Epochs = 300		
Nber. of convergence	61/100	84/100	93/100	None	2/100	1/100

Table 6. Convergence tests performed for model optimization. Each model was computed 100 times.

6. CONCLUSION

The goal of this article was to present the way to follow to build, based on machine learning approach, a model of Deep Neural Network able to reach the requirements of the nanoscale metrology. In the first part a presentation of methodology has been done. The necessity to use modeled data for the different measurement techniques (OCD, CDSEM, CDAFM & CDSAXS) allow to obtain a large dataset has been explained. Then the design of the DNN has been presented, describing the architecture and the different parameters within the network. The two last part presented the performances that can be reached by the DNN model and the ways to follow to optimize the Network architecture. The network parameters and hyperparameters have been tested and their influence on the network mainly in terms on convergence have been assessed. The different studies carried out in this work show that the NN tends to absorb the huge errors that could appear on one (or more) parameter. Then it confirms the robustness on results generated by the NN. On the convergence, it shows that there is a balance to find between the number of data used and the number of epochs required.
The last study on architecture shows that in the configuration on this study the model with 2 hidden layers and 5 neurons per layer gives the best results in terms of convergence rate.
This process must be repeated if the configurations of parameters change. The next steps of the work will be to develop physical model of measurement techniques to train and validate the network with a high accuracy. Then the experiments currently done on different tools could be operated to improve the metrology results.

Acknowledgment:

The French RENATECH network supported this work
This work is carried out within the framework of the Electronic Component Systems for European Leadership Joint Undertaking (ECSEL) under grant agreement No 826589—MADEin4

7. REFERENCES

[1] F. Castanedo, « A Review of Data Fusion Techniques », *Sci. World J.*, vol. 2013, p. 1‑19, 2013

[2] J. Hazart et al. "Data fusion for CD metrology: heterogeneous hybridization of scatterometry, CDSEM, and AFM data", Proc. SPIE 9050, April 2014, https://doi.org/10.1117/12.2046484

[3] N. Figueiro, "Hybrid Metrology for dimension control in lithography", PhD thesis of Grenoble Alpes University, defended in October 2017

[4] J. Reche et al. "Programmed line width roughness metrology by multitechniques approach" J. of Micro/Nanolithography, MEMS, and MOEMS, 17(4), 041005 (2018).

[5] Keras Simple. flexible. powerful. URL https://keras.io/

[6] USENIX (ed) 2016 12th USENIX Symposium on Operating Systems Designand Implementation (OSDI 16) ISBN ISBN 978-1-931971-33-1 URL https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[7] Scikit-Learn scikit-learn: Machine learning in python URL https://scikit-learn.org/stable/

[8] Pandas URL https://pandas.pydata.org/

[9] NIST/SEMATECH e-Handbook of Statistical Methods, http://www.itl.nist.gov/div898/handbook/, 2017.

[10] I. Gereige et al. "Optimal architecture of a neural network for a high precision in ellipsometric scatterometry", Proc SPIE 6648, Sept 2007, https://doi.org/10.1117/12.734278

[11] I. El Kalyoubi, "Développement de la technique de scattérométrie neuronale dynamique », PhD thesis of Grenoble Alpes University, defended in June 2015.

[12] Brownlee J 2016 Support vector machines for machine learning URL https://machinelearningmastery.com/support-vector-machines-for-machine-learning/

[13] Reyes K 2018 Kernel methods, Tree-based methods, ensemble learning and dimensionality reduction, Pennsylvania University mRS Tutorial GI01 URL https://goo.gl/cXQPtx