Multi-branch neural network for hybrid metrology improvement

P. Digraci, M. Besacier, P. Gergaud, G. Rademaker, J. Reche

P. Digraci, M. Besacier, P. Gergaud, G. Rademaker, J. Reche, "Multi-branch neural network for hybrid metrology improvement," Proc. SPIE 12053, Metrology, Inspection, and Process Control XXXVI, 120530W (26 May 2022); doi: 10.1117/12.2612798

SPIE.

Event: SPIE Advanced Lithography + Patterning, 2022, San Jose, California, United States

Multi-branch neural network for hybrid metrology improvement.

P. Digraci[ab], M. Besacier[a], P. Gergaud[b] , G. Rademaker[b], J. Reche[b]

[a]Univ. Grenoble Alpes, CNRS, CEA/LETI-Minatec, Grenoble INP, Institute of Engineering and Management University Grenoble Alpes, LTM, Grenoble F-38054, France

[b]Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France

ABSTRACT

In the domain of advanced patterning, and especially at lithography steps achieve very small sizes becomes more and more crucial. This induces measurement challenges and thus requiring the development of new, precise and robust metrology techniques. To overcome the limited constraints of different techniques, one of the most promising approaches is hybrid metrology. It consists in gathering several metrology techniques to measure all the geometrical parameters which are processed them by an algorithm (mainly machine learning algorithm). This work stands out by using for deep learning a multi-branch neural network to increase the precision of predicts. With a particular attention made to the dataset generation and specific settings for each branch, we developed the potential of this approach which increase the precision of predicts.

Keywords: Hybrid dimensional metrology, Critical dimensions, Advanced lithography, Nanoscale characterization, Data fusion, Artificial intelligence, Neural Network, Multi-branch architecture.

1. INTRODUCTION

The industry of semiconductors is rapidly evolving, developing new techniques and proposing new technology nodes on a regular basis. A part of these breakthroughs reduce the size of the features requiring a stricter dimension control. As the features of devices become smaller, we need to develop metrological techniques which reduce measurement uncertainties and meets the needs of the semiconductor industry. This requires more intelligent metrological approaches such as hybrid metrology [1].

Hybrid metrology follows the concept of the data fusion [2]. It consists in the use of different metrological techniques and merging the different information obtained. This allows to combine the strengths of each one and to attenuate their deficiencies [3]. The aim is to increase the precision of measurements. As explained in our previous work [6], we used the metrological techniques: Critical Dimension Scanning Electron Microscope (CD-SEM), Atomic Force Microscope (AFM-3D), Optical Critical Dimension (OCD) (scatterometry) and Critical Dimension Small Angle X-ray Scattering (CD-SAXS).

The targets of our study are silicon lines gratings with fixed pitch (p = 100 nm, 113 nm, 128 nm…) characterized with 3 dimensional parameters: The critical dimension (CD), height (H) and side wall angle (SWA). To merge every dimensional parameters (DPs) measured by each technique we use an artificial neural network (ANN). It takes these parameters in input to predict in output: CD, H and SWA with higher precision. Our previous work [6] were focus on the development of ANNs for hybrid metrology. In this article, we present a new architecture that increases the precision of prediction. The different stages of the neural network design are presented.

Metrology, Inspection, and Process Control XXXVI, edited by John C. Robinson, Matthew J. Sendelbach,
Proc. of SPIE Vol. 12053, 120530W · © 2022 SPIE · 0277-786X · doi: 10.1117/12.2612798

In this article, we will see in a first part how to generate a viable dataset. Next, we will deal with the training of our different artificial neural network models. Then we will compare the performances of our different models. Finally, we will open discussion with the creation of a new dataset.

2. METHODOLOGY FOR DATASETS GENERATION

ANNs [4] are computing systems which learn thanks to examples on which we know inputs and output values. They are composed of multiple layers of neurons. Each neuron has its own weights which are modified during training to match the values of the output neurons (predictions) with examples known outputs values. The quality of the prediction depends on the quality of the learning. Restricted by time, we cannot have enough experimental data to train an ANN. Then, we need to generate virtually a dataset. A special attention must be paid in order to obtain interesting and realistic results. The number of sample of our dataset is based on the size of the dataset from our previous work [6] (7500 samples). This size is deliberately high in order to avoid any problems related to a lack of example. Our dataset is split randomly in two parts: a train dataset and a test dataset. The first one is used to train the model, the second is used to evaluate this learning.

$Data_{True}$ Generation

In our methodology we start from generation of known output values which will allow to train the ANN. These values correspond to the true morphological description of different samples, we call them CD_{true}, H_{true}, and SWA_{true} and they form the output dataset "**data true**". We set the ranges of our dimensional parameters, which correspond to the ranges of our experimental samples, as follows:

$$30\ nm < CD < 65\ nm \qquad 20\ nm < H < 100\ nm \qquad 80° < SWA < 90°$$

The true DP values are independent. There are chosen according to the target characteristic. In order to make a dataset, we need to choose the DP value of sample. To avoid any conditioning on our data during the learning process, the DPs of our samples were selected thanks to random uniform generation, as visible Figure 1. Thus, we don't have any correlation between our DPs. These lists (a list for each DP) allow the ANN to better generalize the relationships between the measured values and the values to be predicted.

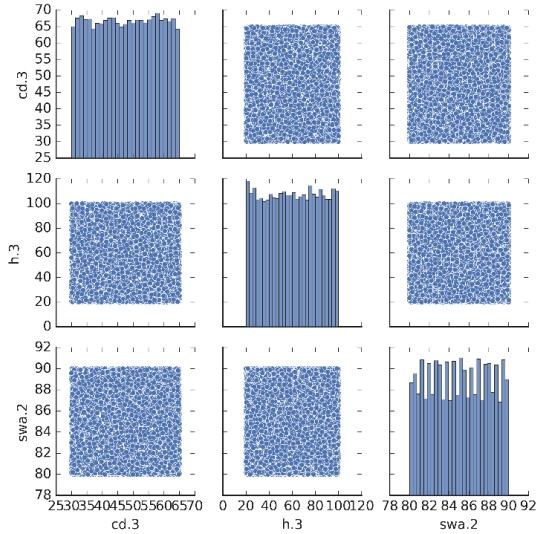

Figure 1. Distribution chart of data true. The graphs on the diagonal represent the distribution of values. The other graphs represent the relationship between the dimensional parameters.

Data$_{virtual}$ Generation

Using the set of data$_{True}$ we generate another set of data by adding some errors, to mimic measurement obtained by each techniques. We call them **"data virtual"**. To add the errors, we use a Gaussian noise distribution. We estimate a realistic parameter of the Gaussian, 3σ = 5% of the true value. Generation is performed through Gaussian distribution with 3σ = 5% of consider parameter. This value is consistent with error which could be obtained during measurement with different techniques. In a first approach we do not include correlation between parameters which can exist for different parameters extract with same technique. We have access to the 3 DPs thank to each technique except for CDSEM technique which we only have access to the CD due to top-down view which given. Thus we obtain for each sample 10 inputs (data$_{virtual}$) created thanks to the 3 outputs (data$_{True}$), these ones are combined to a dataset usable for ANN training as explain in the Figure 2.

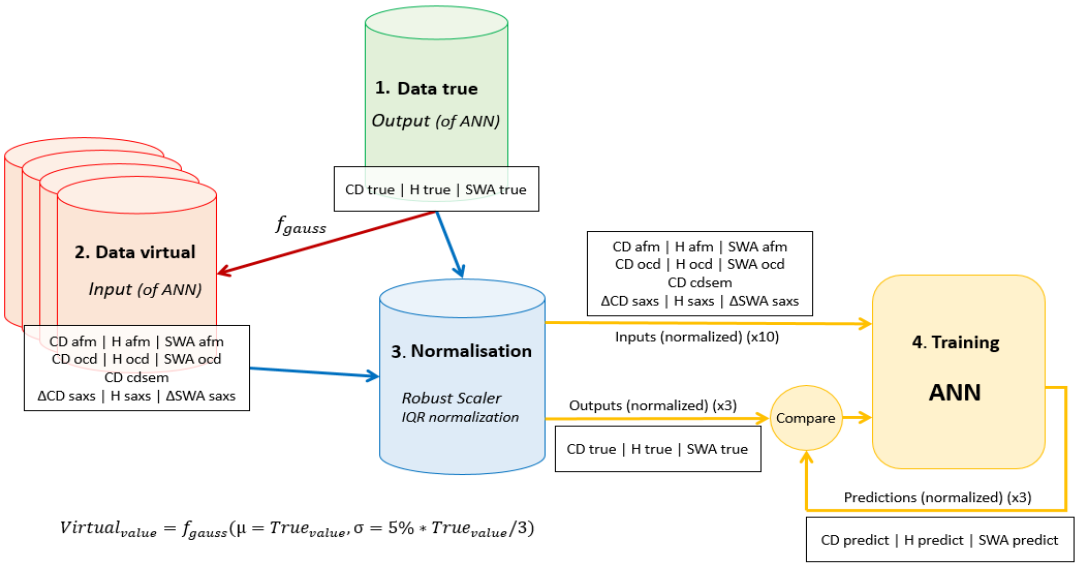

Figure 2. Dataset creation steps

Data Normalization

Normalize datasets is a common requirement for machine learning models [5]. Indeed, the inputs and outputs can have different range of values that can influence the quality of results. As in our previous works [6], we used the InterQuartile Range (IQR) normalization on our dataset. This technique describes as a robust measures of scale [5]. By comparison with the standard normalization which uses mean and standard deviation (stdev) (eq. 1), the IQR normalization (also named Robust Scaler) uses median and the quartiles upper (Q3) and lower (Q1) (eq. 2). That avoids us a significant influence of outliers.

In the next equations, X is a list of all values of the same DP and X_i is the value of the sample i.

$$\boldsymbol{Standard\ Normalization}: \quad \boldsymbol{X_{i\_norm}} = (\boldsymbol{X_i} - \boldsymbol{mean}(\boldsymbol{X}))/\boldsymbol{stdev}(\boldsymbol{X}) \qquad \textbf{(eq. 1)}$$

$$\boldsymbol{Robust\ Scaler}: \quad \boldsymbol{X_{i\_norm}} = (\boldsymbol{X_i} - \boldsymbol{median}(\boldsymbol{X}))/\boldsymbol{IQR}(\boldsymbol{X}) \qquad with\ IQR = Q3 - Q1 \qquad \textbf{(eq. 2)}$$

We use the RobustScaler object from Scikit learn API [14]. It allows to normalise and denormalise the dataset which is loaded by using its methods ("*fit*", "*transform*" and "*inverse_transform*"). Denormalisation is necessary to obtain data with physical meaning.

3. TRAINING OF ARTIFCIAL NEURAL NETWORK

In our previous work [6], we defined several models of ANN fully connected. We study them to optimise their settings and find the best architecture. Our aim is to have as little metrological uncertainty as possible. Every model in this paper is developed in Python by using the Keras API [7] on Tensorflow framework [8].

During the learning process (also called training process) of our ANN, we follow the evolution of the differences between output $data_{True}$ and predictions, this is named the loss function. The ANN needs to learn several time the same dataset to refine its learning. This cycle is call "epoch". At each epoch, 2 loss functions are calculated, the loss function of the train dataset and the loss function of the test dataset. We mainly use the loss function of the test dataset. These values tell us about the ability to generalise the problem. This is in contrast to the value obtained from the training dataset, which is used by the ANNs to optimize themselves. In both cases the loss function is calculated through the Mean Square Error (MSE), describe in eq. 3. Then, the closer the convergence value of the loss function is to 0, the better our learning.

Previous work with classic architecture

We define the classical architecture of an ANN as a feed forward ANN model that has sequentially fully connected layers. This type of model has been used in our previous work [6]. A stable model was found (model 6 of Table 1). It has 11 input neurons, 2 hidden layers with 5 neurons each and 3 output neurons. We used a dataset with 11 inputs (all techniques acquired the 3 DPs except AFM which doesn't measure the SWA) then the results can differ from our actual dataset (presented in the section 2).

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 5 : 11/4/4/3	Model 6 : 11/5/5/3	Model 1 : 11/3/3/3	Model 5 : 11/4/4/3	Model 6 : 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 1. Convergence table of classic architecture depending on the number of epoch and the number of data used. This table comes from the article of our previous work [6]

With architecture model 6 of table 1 these several tests are performed and all show that we cannot improve loss function better than 10^{-2}. One can know that multi-output neural networks which make regressions calculate the loss function of each output and combine them [9] (eq. 3). We call them "primitive" loss functions.

$$\boldsymbol{MSE_{model\ 6} = \frac{1}{3}(MSE_{CD} + MSE_{H} + MSE_{SWA})} \qquad \textbf{(eq. 3)}$$

In order to check the final primitive loss values (after the learning process), we calculate manually the MSE of each output separately with the validation dataset, we obtain:

$MSE_{CD} \approx 10^{-3}$ $MSE_{H} \approx 8*10^{-4}$ $MSE_{SWA} \approx 3*10^{-2}$

According to eq. 3, we understand that there is an important imbalance between each primitive loss values. Then the MSE_{model6} focuses more on the SWA output than the other DPs eq. 4 is obtained. That is a problem of optimization of the CD and H predictions. Indeed, this ANN focus on the SWA predictions.

$$MSE_{model\ 6} \approx \frac{1}{3} * MSE_{SWA} \qquad \textbf{(eq. 4)}$$

Multi-Branch Architecture setup

We can find several articles on multi-branch ANNs [13]. This kind of ANN can provide better results by dividing the problem into several. It follows the divide and conquer principle. More and more multi branch architecture are develop such as ResNeXt [10], Xception [11] or SqueezeNet [12]. Moreover, some papers explain the advantage to use a multi branch on the loss function behaviour. This architecture allows us to have a loss function for each branch and therefore each output. Indeed, we only have one output per branch (Figure 3).

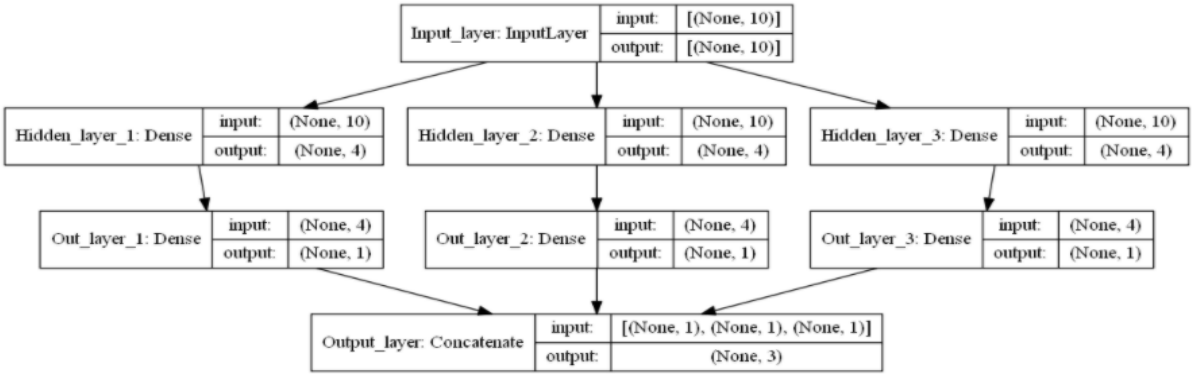

Figure 3. Multi Branch Architecture – Each branch has 1 hidden layer with 4 neurons

In this architecture, each branch can be considered as a sub-ANN that has 10 inputs, a hidden layer and a single output (for one of the DPs). While training this model, if we want a loss function per output, we need to train each branch separately. The outputs of each branch are concatenated (Figure 3).

The choice of the number of hidden layer and the number of neurons on it was made in an empirical manner. The performs are optimal with the minimum of 1 hidden layer with 4 neurons. We can have more but this is not necessary, the results are quite similar, so we won't overload our ANN with extra parameters.

The MSEs of the multi-branch model are lower but are quite similar to the primitive MSEs of the model 6 (Figure 4). These are close to $MSE_{CD}=5*10^{-4}$; $MSE_{H}=2*10^{-4}$ and $MSE_{SWA}=2*10^{-2}$. Compared to the MSE value of the CA, we have significantly better results. The loss functions are an important indicator of the quality of the learning process. They are reflected in our predictions. Thus, the MBA provides better results than the CA.

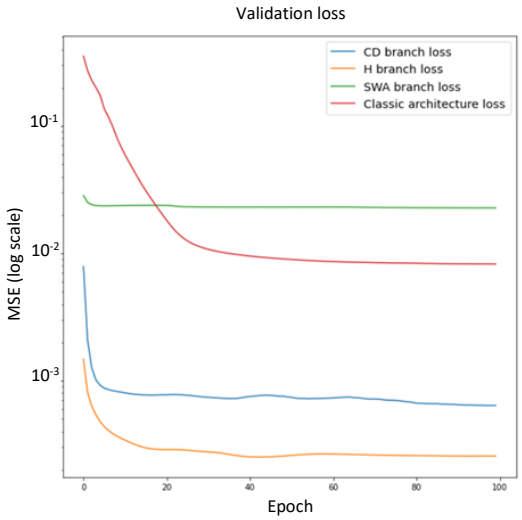

Figure 4. Comparison of the MSE evolution between the "classical architecture" and branches.

4. COMPARISON OF CLASSIC AND MULTIBRANCH ARCHITECTURE

According to our results obtained during the training sessions, we need to evaluate the predictions made by our 2 architectures. In view of the value scales of DPs, we use the relative errors (eq. 1) to make these evaluations.

$$Error_{relative} = \frac{Value_{true} - Value_{predict}}{Value_{true}} * 100 \qquad \textbf{(eq.5)}$$

We represent the histogram of our relative errors to see if our they follow a distribution law. The histogram in Figure 6.a is similar to a normal distribution. To check whether they really follow this distribution, we use the probability plot.

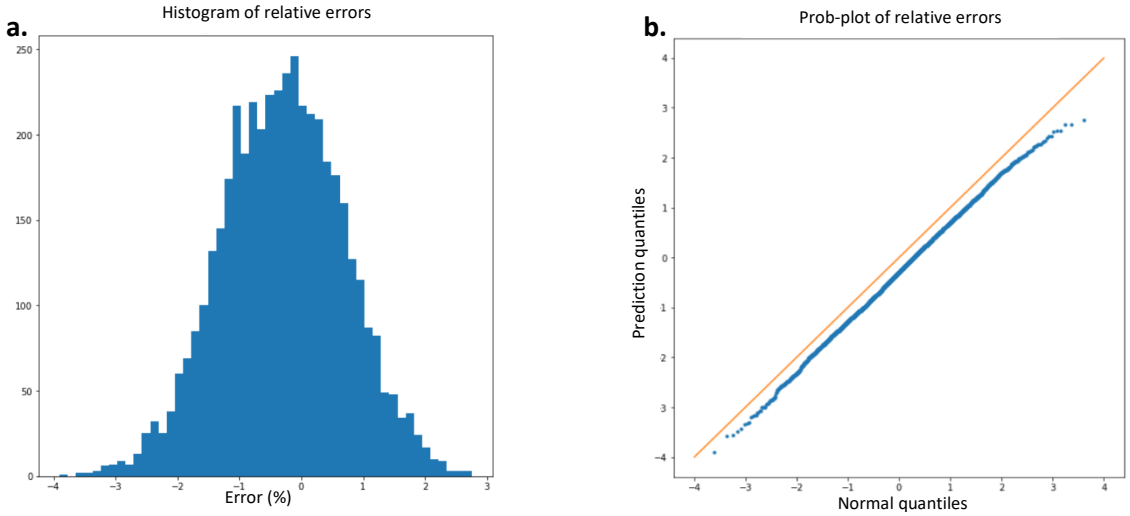

Figure 5. (a) Histogram of relative prediction errors of H. (b) Probality Plot (Prob-Plot) of relative prediction errors of H compare to a normale distribution. We have the same behaviour with CD and SWA.

The probability plot is a graphical technique which is used to compare two datasets. In this case it's used to compare only a single dataset to a reference distribution (normal distribution). Thanks to *stats.probplot* function of Scipy API [15], we have a diagram (Figure 5.b) and R (the square root of the coefficient of determination).

As we can see on this diagram (Figure 5.b), our relative error distribution is close to a normal distribution. Moreover, R = 0.9997, that means we can consider our distribution as a Gaussian distribution but with a non-zero average.

Then we can use an estimator of normal distribution. This estimator is the function *norm.fit* of Scipy API [15]. It finds the best parameters to match with our distribution. On the Figure 6.a, we have our distribution (in red) and its estimated normal distribution (in blue). Thanks to this estimation, we obtain the values of the parameters of this normal distribution. Then, we have an estimation of the prediction uncertainties.

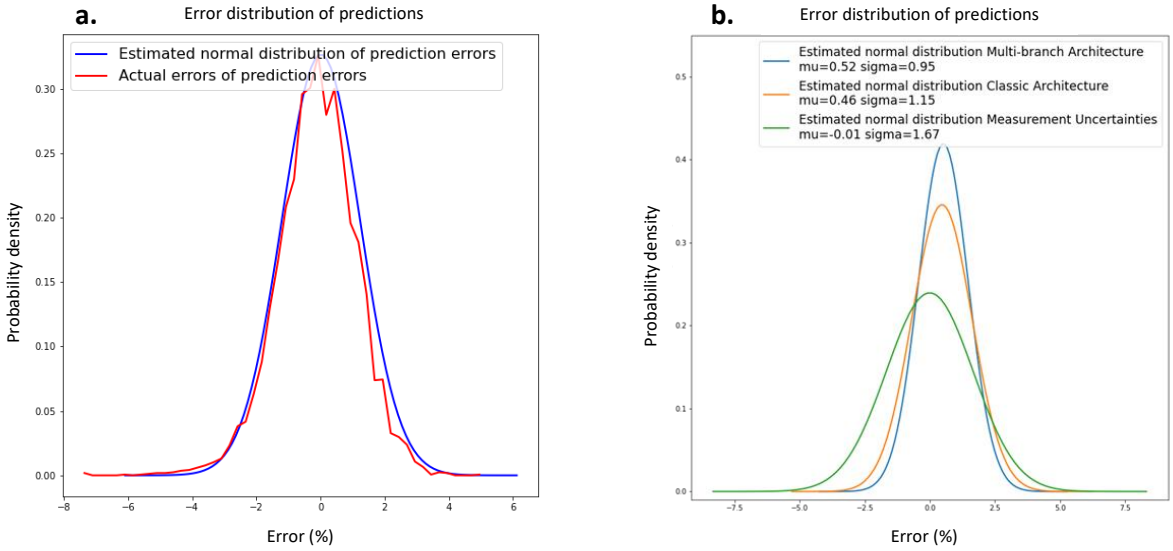

Figure 6. (a) Error distribution of CD prediction of the model 6. (b) Probability density of the estimated normal distributions of prediction errors and metro errors focused on CD, (size of the training dataset = 1000).

On the Figure 6.b, we can see the results of our different architectures. In input, we have 5% (for 3σ) of uncertainties but in output, we have 3.45% (for 3σ) of uncertainties. Then the classic architecture reduces the uncertainty of CD by ≈30%. In comparison, the multi-branch architecture reduces the uncertainty of CD by ≈40%. As CD, the other DPs from the model 6 was significantly lower than measurement uncertainties. In the same way, the MBA decreases the uncertainties. Its predicts are more precise than the model 6 (Table 2).

Table 2. Relative errors of the classic architecture (model 6) and the multi-branch architecture.

Errors (3σ)	CD	H	SWA
Classic Architecture	3,5%	3,8%	2,9%
Multi-Branch Architecture	2,9%	3,0%	2,6%

Then, the multi-branch architecture provides better results than a classic one. The uncertainties of predictions are reduced by about 20% for CD and H and 10% for SWA. This is specific to our problem which is to do 3 regressions on independent (output) data. Nevertheless, this architecture has more parameters (147 compared to 103 with the model 6) and needs about 3 times more time to be train. These disadvantages do not outweigh the advantages. On the contrary, being able to train each branch separately allows a re-training of only those branches that had a learning problem (e.g. a non-convergence).

In view of the performance of the MBA compared to a CA and considering our objective to be as accurate as possible, we decided to use it in our future work.

5. NEW DATASET POSSIBILITIES

Until now, we used the virtual dataset which don't represent correctly the measurement uncertainties. To go further, we need a new dataset representing these uncertainties in a more realistic way. Because of cost and time, we can't make a dataset of experimental sample.

Another way is to use simulators. Indeed, simulators can provide realistic data. This is similar to use a real metrology technique. Nevertheless, they need time to generate samples then we need to estimate the minimum number of sample in our dataset to provide optimal results.

Minimum size of dataset

Thanks to our previous research, we find an optimal architecture (MBA). We use it to estimate the minimum size of dataset to converge to the optimal value. The loss function is an indicator of the quality of the learning process.

We estimated that we needed about 600 samples for training and a few hundred more for testing. As we can see on the Figure 7Figure 7, with 600 data we can reach the minimal value of convergence despite a higher number of epoch to converge. To these samples, we add 100 samples to have enough number of data to test the MBA training. Then we chose to fix the number of samples in our new dataset at 700. This number may change depending on future results. As before, we use the value of our $Data_{True}$ thanks to a uniform distribution

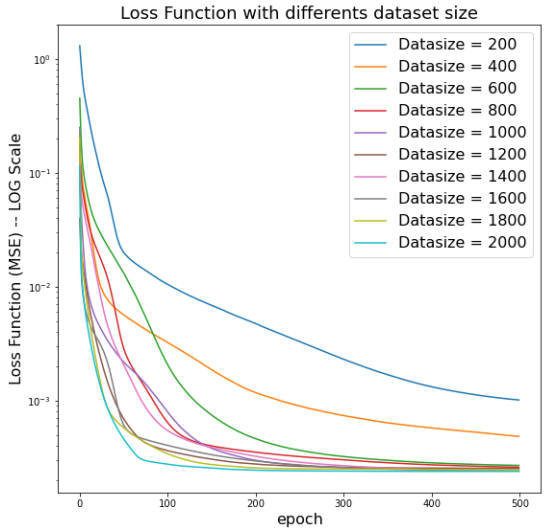

Figure 7. Train loss function of the H branch (1 hidden layer with 4 neurons) with different number of data for the training – the dataset used is the dataset Virtual

Simulations

Simulators are software which reproduce the behaviour of the metrology techniques. Today we already developed some of them. The following part showing simulations obtained and them used in the DP approach.

First, we need to define a geometric model from $Data_{True}$, which is the closest as possible to reality. The one chosen includes corner rounding on the top and bottom of our trapeze (Figure 8.a), that corresponds to experimental samples that we can have.

Every simulator uses the same model. Thanks to this one, it is possible to generate raw data (Figure 8.b) as during measurement for each kind of technique. After it this data are processed as real experimental data to get the DP values.

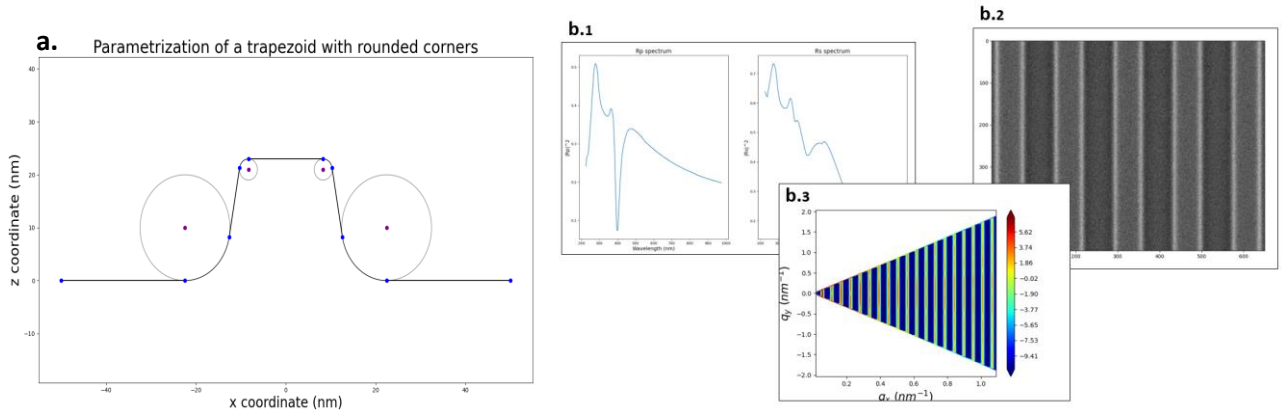

Figure 8. (a) Geometric model with corner rounding of our lines. (b.1) Spectra from OCD simulator, (b.2) image from CDSEM simulator and (b.3) spectrum from SAXS of this model.

Today, the main shortcoming of simulators is that they provide ideal raw data. Indeed, the spectrums, images and other kind of raw data were generating without noise or metrological uncertainties. In future, these two last point will be added to the simulator to obtained DP value after extraction as closer as possible of real experimental data and train ANN on it. This new dataset, called **Data$_{Simulate}$**, is still under development and it will replace input Data$_{True}$.

6. CONCLUSION

The hybrid metrology is a promising approach to meet the expectations of future industries. In this article, we present the data fusion by artificial neural network part of this approach. We focus on 4 metrology techniques (AFM, OCD, CDSEM, SAXS) which are their strengths and deficits.

Due to time and cost considerations, we could not use experimental data to train an ANN. Thus we show on this paper the possibility to generate artificial data set. We highlight that some points must be taken in consideration as distribution of this data set. Moreover, we discuss about limit of conventional architectures when they have multiple independent outputs. Indeed, the loss function used to drive our ANNs is calculated from the "primitive" loss functions of each output. We proposed in this article therefore the multi-branch architecture. We demonstrate that it avoids the problem of loss functions and provides better results.

Then the next step in our work is to generate a new data set using simulators that we start to present. They can provide much more realistic data. These data will be similar to the experimental data. Trained with realistic data, our future model will be able to process experimental data while reducing its measurement uncertainties.

7. ACKNOWLEDGEMENT

We acknowledge the support of European commission, French State and Auvergne-Rhône Alpes region through the funding of ECSEL project MadeIn4 part of IPCEI (Important project of commun european interest) microelectronics and French Nano2022 program.

8. REFERENCES

[1] Reche J., “Nouvelle méthodologie hybride pour la mesure de rugosité sub-nanométrique”, Thesis in nano electronics and nano technology, Grenoble Alpes Université (2019)

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

[3] J. Hazart et al. “Data fusion for CD metrology: heterogeneous hybridization of scatterometry, CDSEM, and AFM data”, Proc. SPIE 9050, April 2014

[4] Dreyfus G., Personnaz L., Toulouse G., “Perceptrons, Past and Present”, Enciclopedia Italiana, In press (1997)

[5] Kaltenbach, Hans-Michael (2012). A concise guide to statistics. Heidelberg: Springer. ISBN 978-3-642-23502-3.

[6] L. Penlap Woguia et al. “Data fusion by artificial neural network for hybrid metrology development”, Proc. of SPIE Vol. 11611, March 2021

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

[8] TensorFlow Core. URL https://www.tensorflow.org/api_docs/python/tf

[9] TensorFlow Losses Mean Square Error. URL https://www.tensorflow.org/api_docs/python/tf/keras/losses/MeanSquaredError

[10] S. Xie et al. “Aggregated residual transformations for deep neural network”, IEEE Conference on Computer Vision and Pattern Recognition, 2017

[11] F. Chollet “Xception : Deep Learning with depthwise separable convolutions”, IEEE Conference on Computer Vision and Pattern Recognition, 2017

[12] F.N. Iandola et al. “Squeezenet : Alexnet-level accuracy with 50x fewer parameters and <0.5 MB model size”, arXiv:1602.07360, 2016

[13] Z. Hongyang et al. “Deep Neural Networks with Multi-Branch Architecture are Less Non-Convex”, arXiv:1806.01845v2, 2018

[14] Scikit Learn Robust Scaler. URL https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

[15] Scipy API. URL http://scipy.github.io/devdocs/dev/