*Proceedings of the 2020 Winter Simulation Conference*
*K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.*

# ENHANCING SCALABILITY OF VIRTUAL METROLOGY: A DEEP LEARNING-BASED APPROACH FOR DOMAIN ADAPTATION

Natalie Gentner
Andreas Kyek
Yao Yang

Infineon Technologies AG
Am Campeon 1-15
85579 Neubiberg, GERMANY

Mattia Carletti
Gian Antonio Susto

Department of Information Engineering
University of Padova
Via Gradenigo 6/B
35131 Padova, ITALY

## ABSTRACT

One of the main challenges in developing Machine Learning-based solutions for Semiconductor Manufacturing is the high number of machines in the production and their differences, even when considering chambers of the same machine; this poses a challenge in the scalability of Machine Learning-based solutions in this context, since the development of chamber-specific models for all equipment in the fab is unsustainable. In this work, we present a domain adaptation approach for Virtual Metrology (VM), one of the most successful Machine Learning-based technology in this context. The approach provides a common VM model for two identical-in-design chambers whose data follow different distributions. The approach is based on Domain-Adversarial Neural Networks and it has the merit of exploiting raw trace data, avoiding the loss of information that typically affects VM modules based on features. The effectiveness of the approach is demonstrated on real-world Etching.

## 1 INTRODUCTION AND RELATED WORK

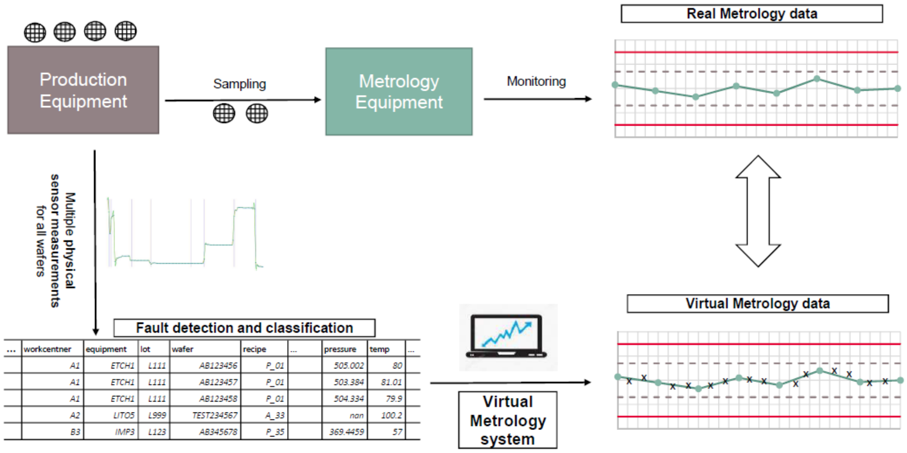
A key factor in semiconductor manufacturing is to create more controlled production environments and to assure quality, speed and stability in production. If an overall standardization strategy is not applicable or only possible to a limited degree, non-identical parts and processes should be aligned and equalized. We refer to this concept as *matching*, a desirable quality under many perspectives: hardware and software configurations, end-of line electrical characterization, tool sensors, process properties, metrology measurements and maintenance actions. At process level, matching is hard to achieve for many reasons, for example product-specific factors like technology and recipe as well as the process dynamics, product mix, process history and equipment status need to be taken into account.

The lack of matching has a profound effect on the scalability and success of Machine Learning-based solution in semiconductor manufacturing. Machine Learning (ML) technologies, like Virtual Metrology, Predictive Maintenance and Fault Detection, have been applied in the past recent years in this context to improve quality and reduce costs. At the present state, ML-based solutions are hardly scalable in semiconductor environment partly due to the lack of matching: for example, even when considering chambers of the same productive machine, a data-driven model developed for one chamber will not be effective for another one. Taken this into account and considering the high number of machines in the shop floor, it is apparent why ML-based solutions are still not consistently adopted even in advanced fabs.

In this work we address the topic of matching and scalability, by considering a particular ML-based technology for process control that is Virtual Metrology (VM). VM aims at estimating metrology quantities that are costly to be measured by means of data-driven algorithms and exploiting the availability of historical

978-1-7281-9499-8/20/$31.00 ©2020 IEEE

data where such measures are performed; VM is formalized as a regression task where sensor data collected from an equipment during a process is treated as input, while the corresponding inline measurements on the wafer are used as output. In Fig. 1 the basic idea and usage of a VM model is illustrated. Since VM enables quality improvement, time effectiveness and cost savings, three critical factors in production, VM has attracted the attention of many as demonstrated by the rich related literature covering almost all production processes. Some of the most interesting aspects in the past recent years (that are, in different ways, related to this work) are reported in the following:



| ... | workcentner | equipment | lot | wafer | recipe | ... | pressure | temp | ... |
|---|---|---|---|---|---|---|---|---|---|
| | A1 | ETCH1 | L111 | AB123456 | P_01 | | 505.002 | 80 | |
| | A1 | ETCH1 | L111 | AB123457 | P_01 | | 503.384 | 81.01 | |
| | A1 | ETCH1 | L111 | AB123458 | P_01 | | 504.334 | 79.9 | |
| | A2 | LITO5 | L999 | TEST234567 | A_33 | | nan | 100.2 | |
| | B3 | IMP3 | L123 | AB345678 | P_35 | | 369.4459 | 57 | |

Figure 1: A typical scheme for a VM technology: VM estimations are typically monitored alongside real metrology measures to widen the monitoring perspective.

1. *Regression Algorithms*: a huge variety of regression approaches have been presented in the literature, from linear methods (Susto and Beghi 2012) to more advanced ML approaches (Lee and Kim 2020). While in early works, 'shallow' ML approaches outperformed neural networks in VM, advancements in Deep Learning (DL) and increased availability of data helped DL become extremely effective in this task both in terms of performance (Maggipinto et al. 2018), (Tsutsui and Matsuzawa 2019), (Wu et al. 2020) and of handling of complex data structures (Maggipinto et al. 2019);
2. *Trace data*: typically VM solutions are developed on key performance indicators or features computed over the equipment sensor signals (Lynn et al. 2009). While this approach is of straightforward applicability, it is sub-optimal, since typically trace data, ie. the full evolution of sensor signals are available: resorting to only statistics of such signals means that a lot of information from the input data is not exploited by the VM module. Some efforts to overcome such limitations, based on modeling directly from the raw sensor data, have been presented: in (Susto et al. 2015) a supervised feature extraction method called SAFE is introduced that provides a regularization approach for estimating scalar output from multiple time-series input; SAFE has the advantage of presenting a closed-form solution and its effectiveness has been proven on a Etching use case. Another paper that presents a strategy to handle directly raw input in VM without resorting to a feature extraction phase is (Park and Kim 2016), where a Fused Lasso-based approach is proposed;
3. *Multiple chamber/machine scenario*: while classically ML-based solutions are developed at a single process/chamber, the real fab scenario call for multi-chamber, multi-machines and multi-stage approaches (Susto et al. 2013). In (Susto et al. 2015) a regularization-based multitask approach is presented to implement VM considering different processes in sequence, while the same scenario is considered in (Wu et al. 2020) exploiting DL architectures.

The aforementioned elements are considered in the present work, where we propose a common VM model for two identical-in-design chambers whose process data follow different distributions; we present a DL-based approach working directly on raw sensor data. More specifically, we resort to a *domain adaption* (Ben-David et al. 2010) approach in combination with an *alignment approach* similar to the one presented in (Farshchian et al. 2019). For the general set up of our alignment system we resort to a DL architecture called Domain Adversarial Neural Net (DANN) (Ganin et al. 2016) that, as explained in the following sections, consists of 3 parts: (i) a feature extractor, (ii) a domain (chamber) discriminator that distinguish which domain is under exam, and (iii) a predictor that provides the VM prediction. Interpretability and enabeling of root cause analysis is one of the main points in production hence we replace the feature extractor that extract latent chamber-unbiased features from the original DANN with an alignment model that shifts one distribution to the other: (Farshchian et al. 2019) presents a DANN related DL architecture for solving a domain adaption task. But instead creating domain independent latent features it uses an autoencoder architecture to map one distribution to the other. This allows in an unsupervised manner to use an already available model on both the original as well as the aligned data and a direct comparison of aligned and original features.

While the work presented here is one of the first example of domain adaption in the context of semiconductor manufacturing, the direct application of such theory in this context is quite underrepresented and only few transfer learning and matching-related work have been presented before: (Kang 2017) proposes a transfer learning method for VM if data shifts between different equipment occur; (Imoto et al. 2018) also uses a transfer learning approach for a convolutional neural network (CNN) architecture for a classification task on wafer defect images: a pre-trained CNN architecture is adopted and retrained for classifying Scanning Electron Microscope (SEM) images with different defect groups. A recent paper from (Chouichi et al. 2020) presents a root cause oriented matching approach of a multiple-chamber equipment by combining multiple multivariate statistical methods: first the source of mismatch related to a metrology measurement is detected by ANOVA, then a best-performing chamber is selected and PLS-DA is used for detecting mismatched chambers based on the selected input data.

The main contributions of this work are summarized as follows:

- A novel VM approach that can cope with multiple systems (chambers or machines) and time-series input data is presented;
- To the best of our knowledge, the presented approach is the first to employ DANN in the context of the regression task of VM and, more generically, in semiconductor manufacturing;
- The effectiveness of the proposed approach is illustrated by showing preliminary results related to variables alignment for a two-chamber etching VM task.

The rest of the paper is organized as follows: Section 2 is dedicated to illustrate the proposed alignment approach; Section 3 is devoted to illustrate the case studies based on synthetic as well as real data. Finally, in Section 4 conclusions are drawn and future works are envisioned.

## 2 METHODOLOGY

In this section we present a DANN-based alignment system, a DL approach that we consider ideal for dealing with VM on process equipment consisting of multiple subtools running in parallel. Since a complete treatise on DL is outside of the scope of this paper we refer the interested readers to (Goodfellow et al. 2016). As argued in Section 1, this architecture works directly on raw sensor signals (trace) respective time-series data.

The main goal of the proposed DANN-based alignment system (depicted in Fig. 2) is to map the data distribution of the second domain to the one of the first domain so the prediction model (*Predictor*) that is trained on the first domain can also be used for the aligned data from the second domain. Therefore we create the *Predictor* itself, an alignment model (*Aligner*) that is used for the mapping after the training is

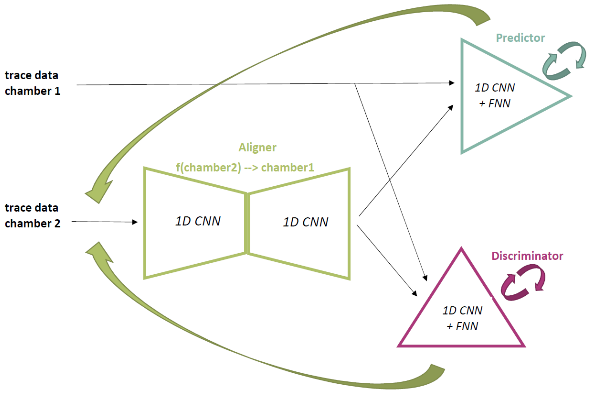


Figure 2: Graphical representation of the proposed VM DANN-based alignment system exploiting input data from two different chambers (=domains). The arrows represent the different feedback during training.

finished and a domain discriminator model (*Discriminator*) that is only used during training in order to assure a successful alignment of the two distributions.

In the following subsections we detail each of the functional blocks in the proposed DANN-based alignment system, its architecture and we discuss implementation and training details.

## 2.1 VM Prediction Model

The baseline for our approach is a good performing VM model for one domain. To this aim we employ Convolutional Neural Networks (CNNs), that are probably the most popular type of DL architectures due to their success in Computer Vision applications. Furthermore CNN proves to be effective in the field of VM ((Lee and Kim 2020), (Maggipinto et al. 2019)). CNNs use filters that extract valuable information from a region of a certain size to detect underlying shapes and topologies from the data it is applied on. Since convolutional filters of any dimension can be applied, CNN are also usable for one-dimensional time series input data: Figure 3 shows a graphical explanation of a one dimensional convolutional layer and a pooling layer with time series data. Compared to other architectures like Long-Short Term Memory (LSTM) or

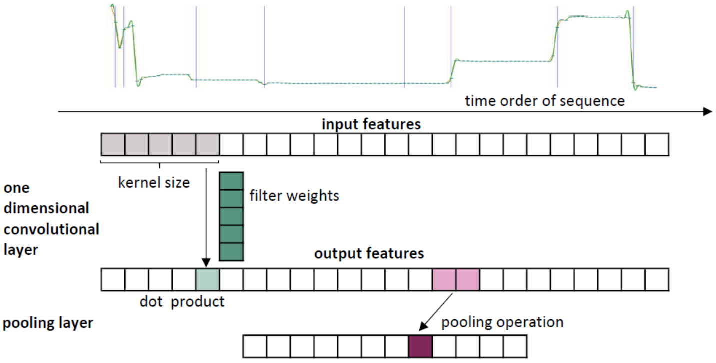


Figure 3: Representation of a one-dimensional convolutional layer followed by a pooling layer applied on time series input features.

Temporal Convolutional Network (TCN) the 1D CNN is easier to implement and to use since no feedback loop is implemented. (Bai et al. 2018) presents an overview and comparison including test studies on different architectures suitable for sequential modeling. We build the *Predictor* with the following modules:

- A convolutional layer suitable for one dimensional input meaning that the kernel only slides along one dimension. The kernel size is representing the width of the kernel and stands for the number of time-steps that are considered at once so it is kind of a window size. The filters are the number of output filters and defines the number of output features you have from this layer;
- An activation function is used to introduce nonlinearity; examples are rectified linear unit (ReLU and Leaky ReLU) or sigmoid (Goodfellow et al. 2016);
- A maxpooling layer reduces the size of the representation, in our case the number of time-steps: we choose this pooling layer so that the more prominent features are kept with the interpretation that peaks for example can have a relevant effect for the VM prediction;
- An upsampling layer is used as reverse operation to pooling in order to increase the representation size. We use this later in the Aligner;
- A batch normalization layer standardizes the input of the layer for each mini-batch during training. Mini-batches are used for example if not all of your data samples can be presented to the algorithm at once due to memory issues. In general this stabilizes the training and therefore reduces the number of training rounds the network needs.

For a visualization of one-dimensional convolutional and pooling layer see Figure 3. We use the convolution layer, the ReLU activation, the pooling layer and the batch normalization layer as one block that can be repeated to reduce the dimension further in order to draw the relevant shapes and dependencies from the data. After those convolutional blocks we introduce in addition: (i) a flattening layer that transforms the multidimensional data to one dimension. This enables the use of multiple fully connected dense layers known from FNN; (ii) multiple fully connected dense layers known from the basic architecture of neural nets. The output layer is one dimensional with only one value depending on our prediction task and sigmoid activation function is used to create values between 0 and 1 since we normalize all data. In a final step we define a suitable loss function like Mean Squared Error (MSE) or Huber loss: a loss function that is quadratic for values smaller than a threshold and linear otherwise and is less sensitive to outliers.

## 2.2 Alignment Model inspired by Autoencoder

An autoencoder consists of two parts: an encoder and a decoder. The encoder maps the input to a lower dimensional latent space without loosing any of the good information that is available in the data. The decoder has the purpose to decompress those latent features back to its original input space. The training happens in an unsupervised manner since the autoencoder should learn to reconstruct its original input. A broad range of layer structures can be used in the autoencoder. We adopt the architecture of an autoencoder model for the *Aligner* and use a 1D CNN based architecture since input as well as output data is time series data for the VM case. The encoder and decoder part each consists similar to the prediction model of:

- Convolutional blocks but with no additional activation function (linear activation);
- Upsampling layers in the decoder part and pooling layer in the encoder part;

In order to have a good initialization we separately pre-train the *Aligner* similar to an autoencoder with the data from the second domain. This is possible if domain one and two have the same underlying data structure. During the training process of the DANN-based alignment system we replace the output data with the data from the first domain so that it learns to map the distribution from the second domain to the one of the first domain instead reproducing its own input. We define a mean quared error loss function.

## 2.3 Alignment System, Training and Regularization

So far the DANN-based alignment system consists of two main parts, the *Predictor* that is able to predict process results for the first domain and the *Aligner* set up as an autoencoder that is able to reproduce the features of the second domain. In order to start the main training process we create the *Discriminator* and then put together all three parts to a multi-stream model for the main training process. The *Discriminator* is built in the same manner as the *Predictor* except no batch normalization layer is applied as recommended by (Gulrajani et al. 2017). The final layer has a one value output with a linear activation function so the discriminator learns to make the distance between its output for domain one and domain two as large as possible. The discriminator set up as well as the training routine follow (Gulrajani et al. 2017) to assure stable training. The final DANN-based alignment system is build in the following way:

- *Domain discriminator*: We connect the *Aligner* with the *Discriminator* via a multi-stream architecture. We train it with two inputs from domain one and aligned input from domain two and one output from the discriminator model. The weights of the *Aligner* are fixed when the *Discriminator* is updated;
- *Domain aligner*: We connect the alignment model via multi-stream with the prediction model and at the same time the discriminator. We use data from domain one and two as two inputs and the discriminator loss as first and the prediction loss as second output. During the update of the domain aligner the weights of the *Predictor* and *Discriminator* are fixed so that only the *Aligner* gets updated.

The training happens in an iterative manner where first the *Domain discriminator* and then the *Domain aligner* gets updated. The *Domain Discriminator* is trained by minimizing the Wasserstein distance while the *Aligner* is trained by minimizing the negative *Discriminator* loss (maximizing the *Discriminator* loss) plus the MSE loss used for the *Predictor*. For details on the training approach, the Wasserstein distance, the gradient penalty regularization and how to prevent misbehavior and vanishing gradients during training see (Gulrajani et al. 2017). The update ratio - the number of times the *Discriminator* compared to the *Aligner* is updated - is set so that none of the models overwhelms the other and is recommended 20:1. Since this kind of optimization is neither linear nor convex this continues until the training reach a saddle point where the *discriminator* performance is close to random guess and the *Predictor* accuracy of the second domain by using its aligned features is as close as possible to the one of the first domain. For training we use the Adam optimizer (see (Goodfellow et al. 2016)).

## 3 CASE STUDY

Here we train the introduced DANN-based alignment architecture on synthetic test data. We analyze normal distributed input features where the two domains are characterized by a shift in the mean values. The input is mapped with a square function to the output space independently of the shift. We increase stepwise the complexity of the synthetic input feature from time-independent features where we use a FNN architecture to time series features where a 1D CNN architecture is exploited. In the second part of chapter 3 we apply the approach to real semiconductor manufacturing data from an Etching process. The Etching process is selected based on availability of production as well as metrology data and popularity concerning other VM research and literature. Besides etching other processes like CVD for example are often considered. Nevertheless the presented approach is use case and task independent and can be transferred to other use cases. Features that are relevant for the VM predictions and show a clear data shift are aligned exploiting a 1D CNN architecture suitable for raw sensor respectively time series input data.

### 3.1 Synthetic Data

### 3.1.1 Experiment 1 - Synthetic data without time dependency

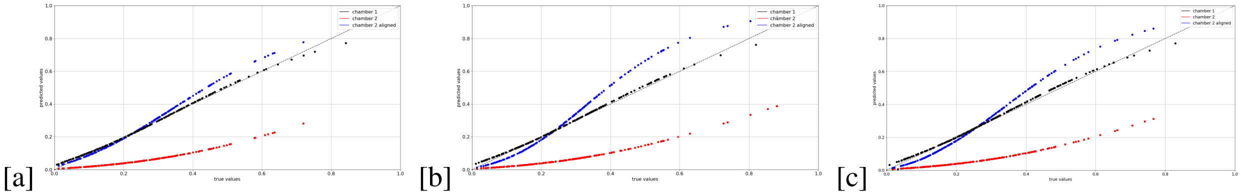
We start with a one dimensional input feature space consisting of two Gaussian distributed subsets $X_1$ and $X_2$. The output space Y is generated via a quadratic mapping. For parameter details see Table 1. We generate 1000 samples from $X_1$ and $X_2$. Then we compute the corresponding $Y$. No noise is added and all features are normalized. *Predictor*, *Aligner*, *Discriminator* are set up as multi-stream FNN:

Table 1: Parameter details of the generated synthetic data for Experiment 1.

| Domain | Distribution | Mean | Std | Output |
|---|---|---|---|---|
| $X_1$ | Gaussian | 0.6 | 0.2 | $y = x^2$ |
| $X_2$ | Gaussian | 0.2 | 0.2 | $y = (x-0.4)^2$ |

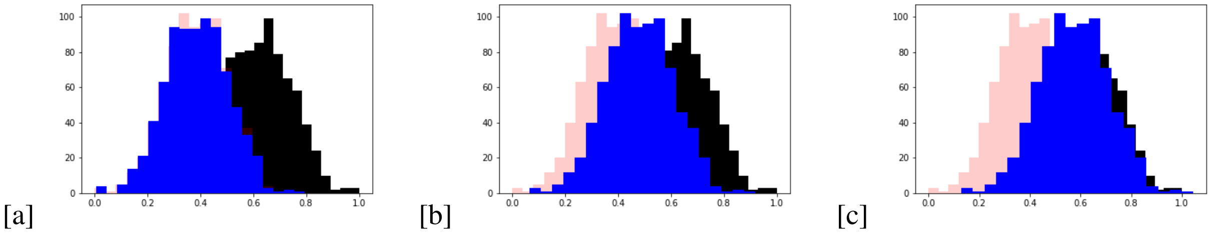
- *Predictor*: 1 dense layers with dimension 4 and a linear activation and an output layer with 1 unit and a sigmoid activation function;
- *Aligner*: 1 dense layers with unit size 4 plus ReLU activation function and an output layer with dimension 1 and linear activation;
- *Discriminator*: 2 dense layers with ReLU respective linear activation function.

We train the *Predictor* with data from $X_1$ and the corresponding $Y$ using Huber loss. Using the trained model for samples from $X_2$ we see that the prediction accuracy is suffering from the shift of the mean value in the input data and therefore shows a ten times higher error. For a comparison of predicted versus true values of the test data for $X_1$, $X_2$ and $X_2$ after the alignment see Figure 4. We pre-train the *Aligner* with $X_2$ and MSE loss to have a good initialization for the *Domain aligner* needed to shift the distribution of $X_2$ to the one from $X_1$. We create the *Domain discriminator* and *Domain aligner* and follow the recommendations about hyperparameter settings from (Gulrajani et al. 2017): so the training ratio is 20:1, gradient penalty weight is 10. Figure 5 shows the progress of the data shift during successful training.



[a] [b] [c]

Figure 4: Graphical representation of true vs. predicted Y for synthetic test data for $X_1$ (black), $X_2$ (red) and $X_2$ after alignment (blue) for [a] stationary data [b] constant time series and [c] time series with jump.



[a] [b] [c]

Figure 5: Stepwise shift (blue) during training procedure of the normal distributed training data from $X_2$ (red) to $X_1$ (black) after [a] 0 [b] 100 [c] 200 iterations.

### 3.1.2 Experiment 2 - Synthetic data with time dependency

We increase the data complexity by creating time series features with normal distributed mean values. We mimic typical sensor data of an Etching equipment. The median length of a time series from Etching is 118 hence we use this as reference length for our synthetic data. We introduce two different shapes of time series data: the first is a constant value for all 118 data points, the second time series shape has a jump in the middle and the values before and after the jump are shifted by $+/-0.1$. Before and after the jump, the time series is again constant. The mean values of time series samples are again distributed as in Experiment 1 (described in 1). The output Y is generated via the quadratic mapping using the mean value. *Predictor*, *Aligner*, *Discriminator* are set up as multi-stream 1D CNN:

- *Predictor*: 1 convolutional block with with filter and kernel size 7, causal padding, stride 1 and linear activation function, a max-pooling layer with size 4 and 2 dense layers after flattening with output size 4 and 1 linear activation for the first dense and sigmoid for the last layer;
- *Aligner*: 1 conv. block each for decoder and encoder with 1 filter, kernel size 3, causal padding, stride 1 and linear activation function; maxpooling respective upsampling is done with factor 2;
- *Discriminator*: see *Predictor* but with LeakyReLU in the convolutional block and linear activation function in the output layer.

We train the *Predictor* with data from $X_1$ and the corresponding $Y$. Using the *Predictor* on $X_2$ shows low accuracy due to the shift of the mean value in the input data. For a comparison of predicted versus true values of the test data before and after the DANN-based alignment see Figure 4. The *Aligner* is pre-trained with data from $X_1$. *Domain discriminator* and *Domain aligner* are put together and trained following the recommendations about hyperparameter settings from (Gulrajani et al. 2017) with training ratio 20:1 and gradient penalty weight 10. Figure 6 shows the progress of the data shift during training for constant as well as constant with jump time series data.

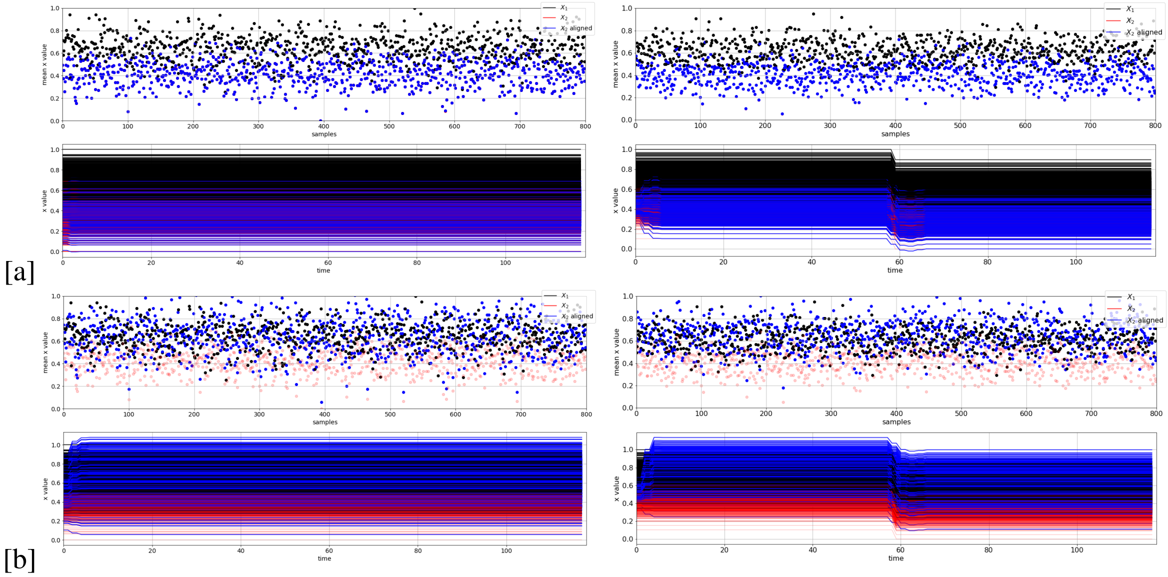


Figure 6: Alignment (blue) during training of synthetic time series data with constant value (left side) and constant with jump (right side) from $X_2$ (red) to $X_1$ (black) after [a] 0 and [b] 200 iterations. Top graph: mean values of all samples; bottom graph: time series samples.

## 3.2 Real Semiconductor Manufacturing Data

To validate the proposed methodology in a real environment, physical sensor measurements of a chamber etching equipment as well as the corresponding metrology measurements on the product after the process are collected and analyzed. Multiple VM measurements are available besides critical dimension (CD) and layer thickness (LT) (for a visualization of these measurements see Figure 7). We choose the latter quantity as target of our VM model since LT shows high correlations to the etching process itself whereas other measurements can be highly influenced by upstream processes, input features that are not available in our case and the performance and the measurement accuracy of the metrology tool itself.

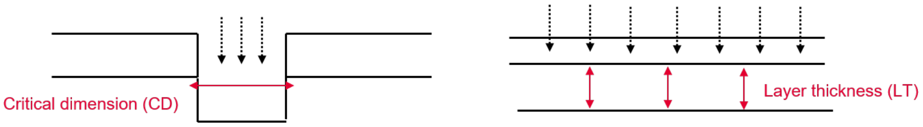


Figure 7: Metrology measurements done after an Etching process: The layer thickness (right) measures the dept of a layer after the process, the critical dimension (left) describes the width or height of a gate.

### 3.2.1 Experiment 3 - Real semiconductor manufacturing data with time dependency

We select one parameter value due to very different target specifications of different metrology parameters. This leads to a reduced input data set with focus on a specific product group and the corresponding recipe. We select data of two chambers for a time period of one year where no other domain shift within one chamber for example on the time axis is detected. First we identify the chamber mismatch of two chambers running in parallel on the same Etching tool. A one-to-one comparison of the features of the two chambers show some clear shifts of some sensors measurements while others do not differ at all. Those shifted physical sizes can be interpreted as compensator of an underlying chamber mismatch - identical-in-design but not in execution - and therefore defines the chamber mismatch related to a process we want to present a solution for in this paper. We define chamber 1 as reference chamber since it provides the highest number of samples for the selected layer thickness parameter in the selected time period. Similar to Experiment 1 and 2 the output data of both chambers is following the same distribution since no statistical significant difference can be detected (see boxplot graphs in Figure 8).

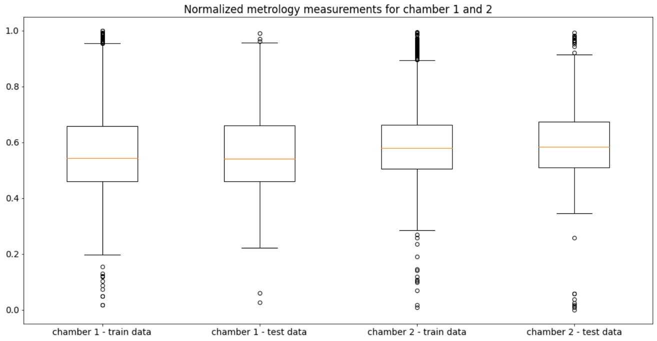


Figure 8: Boxplot graphs of normalized Metrology measurements from both chambers.

Some preprocessing steps are applied on the raw sensor data: we remove outliers as well as constant features for example gas flows that are not used in this recipe. Remaining NaN values are replaced with 0 since features are normalized between 0 and 1. Etching is a multi-step process where not all steps are time restricted but answer on available end point detection sensors: this leads to time series data with unequal duration. Since we see that samples are clustered regarding their time series length, we select the most frequent length respective the median length as reference. Runs with 1-3 additional data points are cut off at the end and runs with 1-3 less points are extended by adding zeros at the end (alternative repeat last value instead of 0). Longer or shorter time series are marked as outlier and removed. We do not align each single measurement as well as single steps. Since we use a 1d CNN architecture with filter size 7 those kind of misalignment are negligible. All input features are normalized over both domains.

We use data from three physical sensor measurements after preprocessing in the alignment process. The 3 features are checked by process experts hence are important and influential for the process result and show a clear shift between the two selected chambers. (see original sensor measurements of chamber 1 and 2 in Figure 9). The input data from both chambers is split up in two subsets, subset 1 includes the

3 selected features and in case of chamber 2 is used as input for the *Aligner*. Subset 2 contains the rest of the available features. The *Discriminator* gets the aligned subset 1 from chamber 2 as well as the subset 1 from chamber 1 as input. Subset 2 from chamber 2 is redirected directly to the output of the *Aligner* where the aligned subset 1 and subset 2 from chamber 2 are merged again in order to be used as input for for the *Predictor*. The 3 parts of the DANN-based alignment system are set as follow:

- *Predictor*: 1 conv. block with 32 output filter and kernel size 7, a ReLU activation function, a maxpool layer of 4 and a batch norm layer; a flattening layer is applied, 1 dense layer with output size 16, a batch norm layer. The output layer is a dense layer with a sigmoid activation function;
- *Aligner*: 2 convolutional blocks each for decoder and encoder; the first encoder block has a convolutional layer with filter and kernel size 7, a causal padding, slide size 1, a linear activation function and a maxpooling layer of size 4. The second convolutional block has the same properties as the first one but a maxpooling size of 3. For the decoder we start with a convolutional block with kernel and filter size 7, a causal padding and slide 1. The upsampling layer has a factor of 8. The second convolutional block of the decoder has 3 output filters, a kernel size of 14, valid padding and slide 1 and an upsampling layer with factor 2 to match the original input size. Pooling respective upsampling sizes are selected according to the length of 118 of the time series samples.
- *Discriminator*: 1 concolutional block with filter size 2, kernel size 7, LeakyReLU activation function and a maxpooling of factor 4, a flattening layer, a dense layer with output size 4 and a LeakyReLU activation function. The output layer has one dimension and a linear activation.

For training the *Predictor* with data from chamber 1 we use a Huber loss function and the Adam optimizer with reduced learning rate of a factor $10^2$. The *Aligner* is pre-trained with data from chamber 2. After setting up the *Domain discriminator* and the *Domain aligner* we follow again the recommendations from (Gulrajani et al. 2017). Figure 9 shows euclidien barycenter averages over test samples for 3 features from chamber 2 before and after the alignment. The results are comparable to the ones we see in literature (see (Farshchian et al. 2019) and (Ganin et al. 2016)) and indicate an successful alignment of all three features.

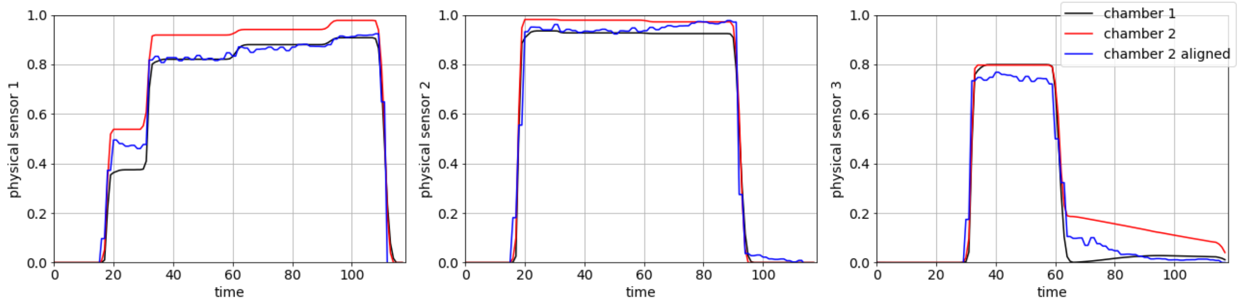


Figure 9: Graphical representation of raw sensor input data before and after the training procedure. The plots show the euclidean barycenter average of all samples in the test data.

## 4 CONCLUSIONS AND FURTHER RESEARCH

The paper presents a matching approach that enables supervised modeling with input data coming from multiple domains, like different chambers or machines; the approach exploits DL architectures, namely DANNs, and it is able to deal directly with time-series input data. The DANN-based alignment system is not only able to deal with the data complexity well known from semiconductor manufacturing but also presents a scalable VM approach in a non-standardized production environment.

The approach is tested on synthetic as well as a real semiconductor manufacturing data set; in all experiments we show that we are able to map one domain to the other which can enable the usage of already existing prediction models to multiple domains. For the real word use case we are able to align physical sensor measurements in their original input space and therefore open the opportunity for a direct feature comparison including physical properties and equipment behavior.

Future research focus on an extended training for stability tests, evaluation and comparison to other approaches: in particular, we are working in translating the preliminary results reported here in the context of VM metrics, by proving a comparison in terms of prediction accuracy of the proposed approach versus standard literature VM approaches. In addition a benchmarking against procedures considering only most important process steps, steps as context input feature or separate models for each step are considered in order to adapt to etching specific time series evolution and properties. The theoretical description of the model, the optimization problem as well as the training approach are added and used for detailed explanation and further discussion. Moreover, since the proposed method provides interpretable results in terms of variable alignment, we aim at evaluating the impact of our approach in terms of limit violations and control charts, root cause analysis related to mismatch, misbehavior and health status.

One direction of development would be in the consideration of higher data complexity by for example extending the number of features as well as considering different product groups, recipes and measurements. Besides VM, we plan also to increase the variety of applications for example to classification tasks, like fault classification. Another direction is to apply the proposed method to semi-supervised problems where labels are partly not available, include long-term time dependencies.

## ACKNOWLEDGMENTS

Infineon Technologies AG is gratefully acknowledged for financing this research activity. Part of this work was supported by MIUR (Italian Minister for Education) under the initiative ”Departments of Excellence” (Law 232/2016).

## REFERENCES

Bai, S., J. Z. Kolter, and V. Koltun. 2018. “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling”. *ArXiv* abs/1803.01271.

Ben-David, S., J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan. 2010, 05. “A theory of learning from different domains”. *Machine Learning* 79:151–175.

Chouichi, A., J. Blue, C. Yugma, and F. Pasqualini. 2020. “Chamber-to-Chamber Discrepancy Detection in Semiconductor Manufacturing”. *IEEE Transactions on Semiconductor Manufacturing* 33(1):86–95.

Farshchian, A., J. A. Gallego, J. P. Cohen, Y. Bengio, L. E. Miller, and S. A. Solla. 2019. “Adversarial Domain Adaptation for Stable Brain-Machine Interfaces”. *ArXiv* abs/1810.00045.

Ganin, Y., E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. 2016, January. “Domain-Adversarial Training of Neural Networks”. *J. Mach. Learn. Res.* 17(1):2096–2030.

Goodfellow, I., Y. Bengio, and A. Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. 2017. “Improved Training of Wasserstein GANs”. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS’17, 5769–5779. Red Hook, NY, USA: Curran Associates Inc.

Imoto, K., T. Nakai, T. Ike, K. Haruki, and Y. Sato. 2018. “A CNN-based transfer learning method for defect classification in semiconductor manufacturing”. In *2018 International Symposium on Semiconductor Manufacturing (ISSM)*, 1–3. IEEE.

Kang, S. 2017. “On effectiveness of transfer learning approach for neural network-based virtual metrology modeling”. *IEEE Transactions on Semiconductor Manufacturing* 31(1):149–155.

Lee, K. B., and C. O. Kim. 2020. “Recurrent feature-incorporated convolutional neural network for virtual metrology of the chemical mechanical planarization process”. *Journal of Intelligent Manufacturing* 31(1):73–86.

Lynn, S., J. Ringwood, E. Ragnoli, S. McLoone, and N. MacGearailty. 2009. “Virtual metrology for plasma etch using tool variables”. In *2009 IEEE/SEMI Advanced Semiconductor Manufacturing Conference*, 143–148. IEEE.

Maggipinto, M., A. Beghi, S. McLoone, and G. A. Susto. 2019. “DeepVM: A Deep Learning-based approach with automatic feature extraction for 2D input data Virtual Metrology”. *Journal of Process Control* 84:24–34.

Maggipinto, M., M. Terzi, C. Masiero, A. Beghi, and G. A. Susto. 2018. "A computer vision-inspired deep learning architecture for virtual metrology modeling with 2-dimensional data". *IEEE Transactions on Semiconductor Manufacturing* 31(3):376–384.

Park, C., and S. B. Kim. 2016. "Virtual metrology modeling of time-dependent spectroscopic signals by a fused lasso algorithm". *Journal of Process Control* 42:51–58.

Susto, G. A., and A. Beghi. 2012. "Least angle regression for semiconductor manufacturing modeling". In *2012 IEEE International Conference on Control Applications*, 658–663. IEEE.

Susto, G. A., A. B. Johnston, P. G. O'Hara, and S. McLoone. 2013. "Virtual metrology enabled early stage prediction for enhanced control of multi-stage fabrication processes". In *2013 IEEE International Conference on Automation Science and Engineering (CASE)*, 201–206. IEEE.

Susto, G. A., S. Pampuri, A. Schirru, A. Beghi, and G. De Nicolao. 2015. "Multi-step virtual metrology for semiconductor manufacturing: A multilevel and regularization methods-based approach". *Computers & Operations Research* 53:328–337.

Susto, G. A., A. Schirru, S. Pampuri, and S. McLoone. 2015. "Supervised aggregative feature extraction for big data time series regression". *IEEE Transactions on Industrial Informatics* 12(3):1243–1252.

Tsutsui, T., and T. Matsuzawa. 2019. "Virtual Metrology Model Robustness Against Chamber Condition Variation Using Deep Learning". *IEEE Transactions on Semiconductor Manufacturing* 32(4):428–433.

Wu, X., J. Chen, L. Xie, L. L. T. Chan, and C.-I. Chen. 2020. "Development of convolutional neural network based Gaussian process regression to construct a novel probabilistic virtual metrology in multi-stage semiconductor processes". *Control Engineering Practice* 96:104262.

## AUTHOR BIOGRAPHIES

**NATALIE GENTNER** is currently an industrial Ph.D. student with the University of Padova and Infineon Technologies AG (Neubiberg, Germany). She earned her M.Sc in Mathematics from the Technical University in Munich. Her research interests include machine and deep learning, automatization in semiconductor manufacturing, system theory and applied analysis. Her email address is natalie.gentner@studenti.unipd.it / natalie.gentner@infineon.com.

**MATTIA CARLETTI** is currently a Ph.D. student in the Department of Information Engineering and Human Inspired Technology Research Centre at University of Padova. He earned his M.Sc. degree in Automation Engineering from the University of Padova. His research interests include Explainable Machine Learning and Anomaly Detection, with special focus on industry 4.0 and healthcare applications. His e-mail address is mattia.carletti@unipd.it.

**ANDREAS KYEK** is Data Scientist at Infineon Technologies AG (Neubiberg, Germany). He started his career in industry as a unit process development engineer at Infineon in 2001. From the beginning he was involved in Advanced Process Control (APC) and took over the development department for APC in 2007. During this time he also co-initiated the ENIAC funded project IMPROVE and was speaker of the executive board. In the years 2009 and 2010 he was member of the steering committee of the AEC/APC conference Europe. Later he joined a Manufacturing Excellence group, where he got involved in Factory Physics. Today he is heading projects where the usage of Big Data and Artificial Intelligence methods on manufacturing data is investigated. He holds a Ph.D. in Physics from the Technical University in Munich. His email address is andreas.kyek@infineon.com.

**GIAN ANTONIO SUSTO** is currently an Assistant Professor with the University of Padova. He earned his Ph.D. from University of Padova and he held visiting positions at University of California at San Diego, Maynooth University and Infineon Technologies Austria AG. He is co-founder of Statwolf LtD. His current research interests include machine and deep learning, industry 4.0 and natural language processing. Prof. Susto received the IEEE-CASE Best Student Conference Paper Award in 2011, the IEEE/SEMI-ASMC Best Student Paper Award in 2012, and the IEEE-MSC Best Student Paper Award in 2012. He is an Associate Editor of the IEEE Transactions on Semiconductor Manufacturing for the area of Process Modeling. His email address is gianantonio.susto@unipd.it. His website is http://automatica.dei.unipd.it/susto.html.

**YAO YANG** is Data Scientist at Infineon Technologies AG (Neubiberg, Germany). She earned her Ph.D. from the University of Mannheim with OR (operations research) related topics. She started her career as a scientist solving supply chain optimization problems in BASF SE (Ludwigshafen, Germany). After joining Infineon in 2017, she was involved in a large variety of semiconductor supply chain projects and is now focusing on data science topics, applying Big Data and Artificial Intelligence methods in manufacturing. Her email address is yao.yang@infineon.com.