2022 33rd Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC) | 978-1-6654-9487-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ASMC54647.2022.9792514

On Updating a Virtual Metrology Model in Semiconductor Manufacturing via Transfer Learning

Rebecca CLAIN[1], Ikram AZZIZI[2], Valeria BORODIN[1], Agnès ROUSSY[1]
[1]*Mines Saint-Etienne, Univ. Clermont Auvergne* CNRS, UMR 6158 LIMOS, F-42023 Saint-Etienne, France
{rebecca.clain, valeria.borodin, agnes.roussy}@emse.fr
[2]*STMicroelectronics*, 190 Avenue Célestin Coq, F-13106 Rousset, France
ikram.azzizi@st.com

***Abstract*—Manufacturing processes are often subject to drifts over production cycles. This study applies the paradigm of Transfer Learning (TL) to support the updating of a Virtual Metrology (VM) model based on a Convolutional Neural Network (CNN). The VM is applied to a Chemical Mechanical Planarization (CMP) to predict the average material removal rate. Through the prism of a benchmark case study, this paper empirically investigates how transfer learning can improve the updatability of a VM CNN-based model.**

***Index Terms*—Virtual metrology, Semiconductor manufacturing, Chemical Mechanical Planarization, Transfer learning, Convolutional neural network, Model updatability**

I. Introduction

Metrology tools are in charge of ensuring the production quality in manufacturing industries. To overcome time and cost limitations involved by metrology tasks, Virtual Metrology (VM) systems are developed to predict metrology variables based on the context, process and wafer state information. The majority of data-driven virtual metrology systems relies on machine learning performing usually in a convenient way under the assumption that the training and testing data are drawn from the same distribution. This condition is only met for steady environments. In practice, semiconductor manufacturing processes are highly time-varying. They are subject to drifts and shifts over production cycles. A drift refers commonly to the deterioration of process performance due to tool wear, whereas a shift refers to a change in performances due to a maintenance operation (e.g., when consumable parts of the tool are replaced) [1].

In this context, process measurements sampled at different points in time could belong to different distributions. Consequently, the VM model trained on the historical data would fail when applied to new available data. Therefore, to guarantee the viability of a VM system, updatability approaches are developed. According to [2], the moving window is the most commonly used approach to update a VM system. For instance, [3] propose a dynamic moving window scheme refreshing a VM model to enhance the prediction accuracy. Just In Time (JIT) learning is another popular approach, but limited to fast learner. In [4], the authors propose a JIT approach for VM modeling, along with a variable shrinkage and a selection method, which uses a Gaussian process regression. The current paper investigates a Transfer Learning (TL) approach for an efficient and effective update of a VM model. TL leverages the knowledge gained in solving a source task in a source domain, and applies it to a new task and/or new domain. Applied to update a model, TL transfers the knowledge of an historical VM model (before an update) to a new (updated) model trained on the new data available. The TL avoids a time-consuming full retraining of a VM model on the new samples while tracking the potential drifts and shifts. This study is complementary to the work done in [5], where TL has been applied to facilitate and enhance the set-up of a VM system on a new chamber. [5] focused on TL between different processes. The TL has been used to adapt a model developed for one process to another process. In the current study, the inter-process differences are out of the scope. TL is applied to ensure the viability of a VM model for a single process subject to drifts and shifts.

This paper is organized as follows. In Section II, the TL framework is introduced. In Section III, the adopted convolutional neural network is presented. In Section IV, the CMP tool is described. The computational experiments are provided and analyzed in Section V and Section VI. Section VII concludes this paper and provides a number of perspectives.

II. Transfer Learning

The current paper investigates the use of TL for an effective update of a VM model. A description of the mathematical framework of TL can be found in [6]. Let us recall the TL definition.

Definition 1 (Transfer learning [6])**.** *Given a source domain (D_S) and a learning task (T_S), a target domain (D_T) and a learning task (T_T), TL aims to improve the learning capacity*

This paper is conducted in the framework of the project MADEin4, which has received funding from the ECSEL JU (Electronic Components and Systems for European Leadership Joint Undertaking) under grant agreement No 826589. The JU receives support from the European Union's Horizon 2020 research and innovation program and France, Germany, Austria, Italy, Sweden, Netherlands, Belgium, Hungary, Romania and Israel.

of the target predictive function $f_t(\cdot)$ in D_T by using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.

The learning task of a VM model before and after an update is the same, the metrology variable to predict and the feature space are similar. The source task is thus equal to the target task i.e., $T_S = T_T$. However, the domain source (D_S) and the domain target (D_T) are not necessarily identical. In a steady environment, D_S and D_T are equal i.e., $D_S = D_T$: The features of both domains share the same distribution. In a time varying environment, as it is the case in semiconductor manufacturing, D_S and D_T can be different due to shifts or drifts impacting the feature distribution ($D_S \neq D_T$). This study exploits TL as a tool to adapt the model trained in D_S to a new domain D_T.

The TL approach, adopted in this study, includes two main steps: **(i)** Train a source neural network on the source domain (D_S), **(ii)** copy its layers to the target neural network, train on the target domain (D_T). The layers transferred can be left frozen, meaning that the pre-trained weights do not change or fine tuned, meaning the pre-trained weights are used as initialization and then tuned during the training. The latent features shared by both domains are transferred through the frozen layers and the initialization of the weights. The specificity of the target domain is learned via the fine tuning.

III. Convolutional neural network

The adopted TL approach requires a neural network base model. The choice of a Convolutional Neural Network (CNN) is not mandatory and comes from an empirical investigation of the performances of different types of neural networks. The ease of implementation and the relative good performances of CNN have oriented the choice of the baseline neural network. This section introduces the CNN, and presents the used architecture. CNN is a class of artificial neural network used for classification and prediction. Most CNNs include two parts: **(i)** a *feature extraction part* composed of an input layer, convolutional layers and pooling layers, and **(ii)** a *predictor part* composed of fully connected layers. The convolutional layer uses filters, which performs convolution operations on input data to extract high-level features, called *feature maps*. The feature maps are then normalized by an activation function, and/or resized by a pooling layer. The fully connected layers take the results of the feature extraction process to make predictions or classifications. A convolutional neural network can be three, two or one-dimensional (1D). The difference lies in the dimension, where the convolution filter slides. This study employs a 1D-CNN for the prediction task. The proposed CNN architecture is described in Fig. 1. The input dimension is dependent of the data used in this study and is explained in Section V-A.

IV. Chemical-Mechanical Planarization tool

To evaluate the contribution of TL for the VM updatability, we conduct numerical experiments on a benchmark dataset from the PHM data competition in 2016 [7]. Data are derived from a Chemical-Mechanical Planarization (CMP) tool, that

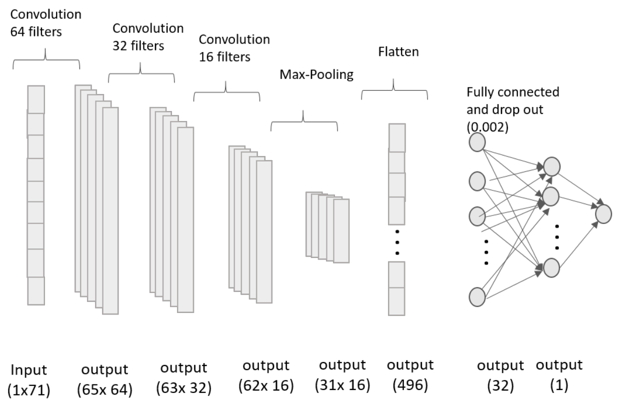

Fig. 1. 1D-CNN architecture

removes undesired material from the surface of wafers via a polishing process. According to [2], CMP is the third most studied area in the VM related literature. Among the CMP-related papers, the topic of tracking and handling drifts and shifts is recurrent. In particular, several studies have highlighted the presence of drifts in the considered dataset from the 2016 PHM data challenge. In [8], the slow drift in this dataset is tracked via an online Bayesian auto-regression exogenous model. The authors of [9] adopted a JIT strategy to track the drifts in the PHM dataset, and continuously learn from the current dataset.

Fig. 2 illustrates the components of a typical CMP tool. A wafer is placed into the wafer carrier and the polishing process starts following a recipe (e.g., set-points for speeds, forces, polish time). The wafer is pressed against the polishing pad and both rotate in the same direction. Meanwhile, the slurry i.e., the liquid containing suspended abrasive components, is spread onto the polishing pad, which generates various chemical and mechanical reactions at the wafer/pad interface. The pad suffers of degradation during the process and needs to be reconditioned by the dresser after the operation. During the polishing process ability of materials, such as the polishing pad, the backing film and the dresser table are diminished. After a certain period, it is necessary to replace them. The material conditions affect the polishing performances, and thus the material removal rate. To track the degradation of the material, usage measurements are collected during the run.

V. Computational experiments

A. Dataset description and preprocessing

The studied dataset consists of the trace signals from various runs of the CMP tool over a period of time. The time series (i.e., trace signal) are related to pressure, slurry flow, rotation and usage of materials. Besides these variables, the metrology measurement is given as an average Material Removal Rate (MRR) for each wafer in the dataset. Fig. 3 displays the time series of the usage variables (e.g., the variables displaying the material usage). The usage variables evolve in time. The variables "Usage of backing film", "Usage of membrane" and

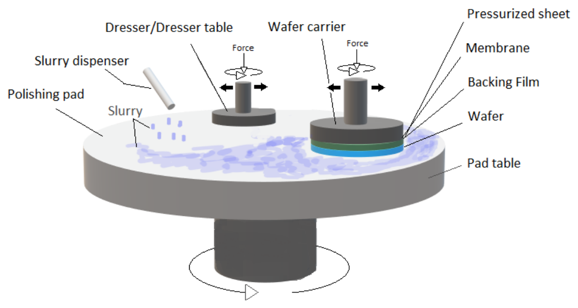

Fig. 2. Ilustration of a CMP tool

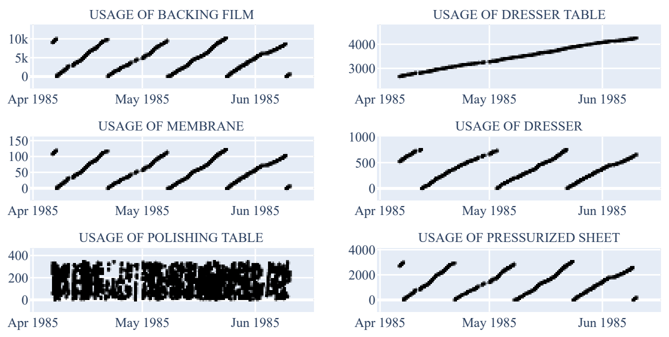

Fig. 3. Time series of Usage variables. The X-axis represents the timestamps and the Y-axis the units of usage measure.

”Usage of pressurized sheet” are highly correlated, and so evolve similarly in time (see Fig. 3). These usage variables increase until reaching a certain point and collapse to zero. The three mentioned materials are related to the wafer carrier (see Fig. 2). It can be inferred, from the time series, they have been replaced four times in the period covering by the benchmark and at the same moment. The dresser has been replaced three times and the dresser table was not changed, its time series growths and never collapses. The time series of the polishing table usage is hardly interpretable.

The provided dataset contains three subgroups corresponding to different processing conditions defined by the chamber ID and the processing stage. These conditions may correspond to different couple of processes and recipes. However, there is not enough context information provided by the benchmark dataset to confirm this hypothesis. This paper investigates the VM model updatability for one processing condition, which corresponds to the wafers passing through three chambers successively and performing a specific stage. The processing condition with the biggest wafer volume has been elected for performing the experiments.

As suggested in [10], the Grubbs’ test is adopted to detect outliers in the dependent variable. To extract and select relevant features, an improved version of the approach proposed in [11] has been used. The final set contains 71 features (see the input size in Fig. 1). All features are normalized.

B. Experimental framework

To investigate the benefits of TL for updating a VM CNN-based model, two scenarios are studied leveraging the studied dataset. In the first scenario, the wafers, ordered by time, are used to create two equal datasets, respectively a source dataset and a target dataset. The wafers related to the source dataset are processed before the wafers of the target dataset. As described in Section V-A, some materials of the CMP tool evolve in time and observe degradation. The source and target domains are likely to be different. This scenario corresponds to a situation, where one needs to update a deprecated model trained on the source domain (source dataset) with upcoming new data (target dataset) under changed equipment conditions. For the second scenario, the wafers belonging to the source and target datasets are chosen randomly. The source and target domains do not have a noticeable difference. This scenario corresponds to a situation, where one needs to update a deprecated model trained on the source domain with upcoming new data. In reality, it is difficult (even impossible) to know with certainty, if one tool was or was not subjected to any drift or shift. However, this hypothetical scenario improves the understanding of the TL capability to enhance the updatability of a VM model. As illustrated in Fig. 4, the data are sorted by time (see upper rectangle in Fig. 4) in Scenario 1. In Scenario 2, data are shuffled (see bottom rectangle in Fig. 4). Four different learning strategies are compared for each scenario (see Table I). The first learning strategy (`Model_0`) evaluates the source model, trained directly and only on the source domain without further training. In the second learning strategy (`Model_1`), a new CNN is trained from scratch on the target dataset only. In the third learning strategy (`Model_2`), a new CNN trained from scratch on both the source and target datasets is tested (the datasets are concatenated to form one dataset). In strategy (`Model_3`), i.e. the TL-based strategy, the source model trained, downstream on the source dataset, is transferred and trained on the target dataset. `Model_3`is slightly different between scenarios. In Scenario 1, the source model is transferred and the convolutional layers are frozen. In Scenario 2, the source model is transferred and none of the layers are frozen. In Scenario 2, the domain source and target are different and so the models necessitate more flexibility to adapt to the target domain. Therefore, for Scenario 2 all the transferred weights are fine-tuned, being included the weights of the convolutional layers. The studied strategies are summarized in Table I.

Both the source and target datasets are divided into two sets: a training set (80%) and a test set (20%). During the training of a model, the training dataset is divided into training and validation sets and the Mean Squared Error (MSE) is computed after each epoch (i.e., an entire pass of the dataset through the neural network). If the MSE does not decrease during 40 epochs consecutively, the training stops. This early stopping approach, aims to avoid the overfitting issue. The learning strategies are evaluated for different split ratio of the target training dataset (from 10% to 100%). `Model_2` is trained

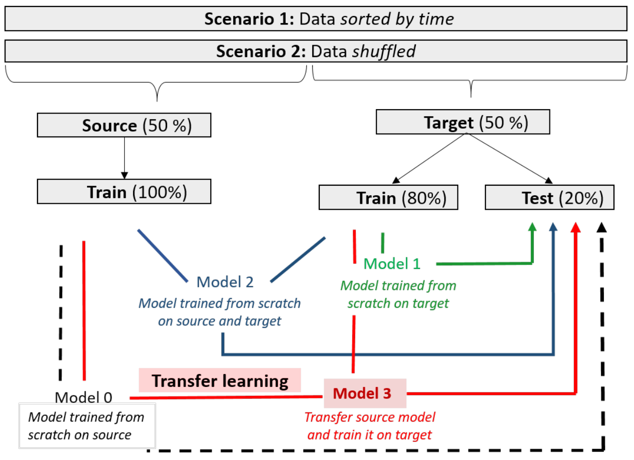
Scenario 1: Data sorted by time
Scenario 2: Data shuffled
Source (50 %)
Target (50 %)
Train (100%)
Train (80%)
Test (20%)
Model 2
Model trained from scratch on source and target
Model 1
Model trained from scratch on target
Transfer learning
Model 0
Model trained from scratch on source
Model 3
Transfer source model and train it on target

Fig. 4. Illustration of investigated learning strategies

on the full source training dataset plus 10% (then plus 20%, plus 30%, etc.) of the target training dataset then evaluated. `Model_3` corresponds to the source model transferred and trained on 10% (then 20%, 30%, etc.) of the target training dataset then evaluated. To ensure robust results, the models have been trained 10 times, and the mean and the standard deviation (σ) of the results have been computed. The learning strategies are evaluated on the test set of the target dataset in terms of average mean squared error (denoted by $\overline{MSE}$) on 10 runs, σ of the MSE on 10 runs, average training CPU on 10 runs and σ of the training CPU on 10 runs.

TABLE I
LEARNING STRATEGIES INVESTIGATED IN SCENARIOS

Model	Notation	Learning strategy
Source model	`Model_0`	Model trained from scratch on the source dataset
Baseline: No TL	`Model_1`	Model trained from scratch on the target dataset
Baseline: No TL	`Model_2`	Model trained from scratch on source and target datasets
TL	`Model_3`	Model trained on the target dataset based on the knowledge learned by `Model_0`

VI. Results and Discussions

In this section, the results associated to Scenario 1 and Scenario 2 are presented and analyzed. Fig. 5 presents the accuracy for the studied models, by varying the size of the target training dataset in Scenario 1 and Scenario 2. The displayed accuracy metric is the $\overline{MSE}$. A lower $\overline{MSE}$ indicates a better accuracy. The $\overline{MSE}$ displayed is the mean over 10 runs and the gray area represents standard deviation, denoted by σ, on 10 runs. Table II displays $\overline{MSE}$ and σ. Fig. 6 illustrates the average training CPU for the studied models, by varying the size of the target training dataset for both scenarios.

Regarding Scenario 1, `Model_2` trained from scratch on both datasets, source and target, performs the worst in terms of accuracy and stability of the prediction. During training, `Model_2` learns features specific to the source domain, which are not relevant to the target domain. It reveals, the source and target domains are, as a matter of fact, different from each other. `Model_1`, trained only on the target dataset, has better performances than `Model_2`. Consequently, the additional input of data from the source domain deteriorates the quality of the prediction. `Model_3` (with TL) dominates other models in terms of $\overline{MSE}$, for almost all the percentages of the target dataset except for 70% and 80%, where `Model_1` obtains a better $\overline{MSE}$. However, for these percentages, σ associated to `Model_1` are, respectively, 5.03 and 7.95 against 0.37 and 0.80 for `Model_3`. This high variability of `Model_1` is also noticeable for other percentages of the target dataset. On average, σ associated to `Model_1` is 5 times superior compared to `Model_3`. In particular, for `Model_1` trained on 100% of the target dataset, its σ explodes and reaches 21.30. It means the MSE of `Model_1` for some of 10 runs are very high and far from the others. For these runs, `Model_1` does not manage to produce reliably consistent results. In other words, `Model_1` once or more over 10 runs, fails to converge while training. A credible hypothesis is the neural network architecture of `Model_1` has to change drastically to be adapted to the target domain, which requires time and engineering tests.

`Model_0` (source model) fails totally on the test set of the target dataset. Its $\overline{MSE}$ is equal to 71.59. Obviously, `Model_0` which has not been trained on the target domain, fails to capture the differences between the source and target domains. The training time of `Model_3` is systematically smaller than the training time of other models. The training time of `Model_3` observes an increasing trend with the volume of the training target dataset, but not linear due to the early stopping effect. However, there is still a trade-off between a low training time (small percentage of the target training dataset) and the accuracy.

Regarding Scenario 2, `Model_3` (with TL) has a more stable prediction and a faster training time than the three other models for all the sizes of the target training dataset. `Model_1` performs the worst, and especially for small percentages of the target training dataset. The model fails to generalize due to the lack of data. Contrary to Scenario 1, `Model_2` performs well and better than `Model_1`. When the source and target domains are close, the additional input of source data represents helpful information and leads to a better accuracy and stability of the prediction. `Model_2` and `Model_3` have close performances in terms of $\overline{MSE}$ on 10 runs, both leveraging the knowledge of the source and the target domain to reach good MSE. However, `Model_3` directly exploits the knowledge present in the source model, saving an important computational cost. The source model reaches a proper accuracy on the test target training dataset even via `Model_2`˙`Model_3` obtains a better accuracy with 50% and more of the target training dataset.

In summary, the experimental results conducted in the framework of Scenario 1, highlight that the source VM CNN-based model would fail over time due to drifts over production cycles. The differences between the source and target domains

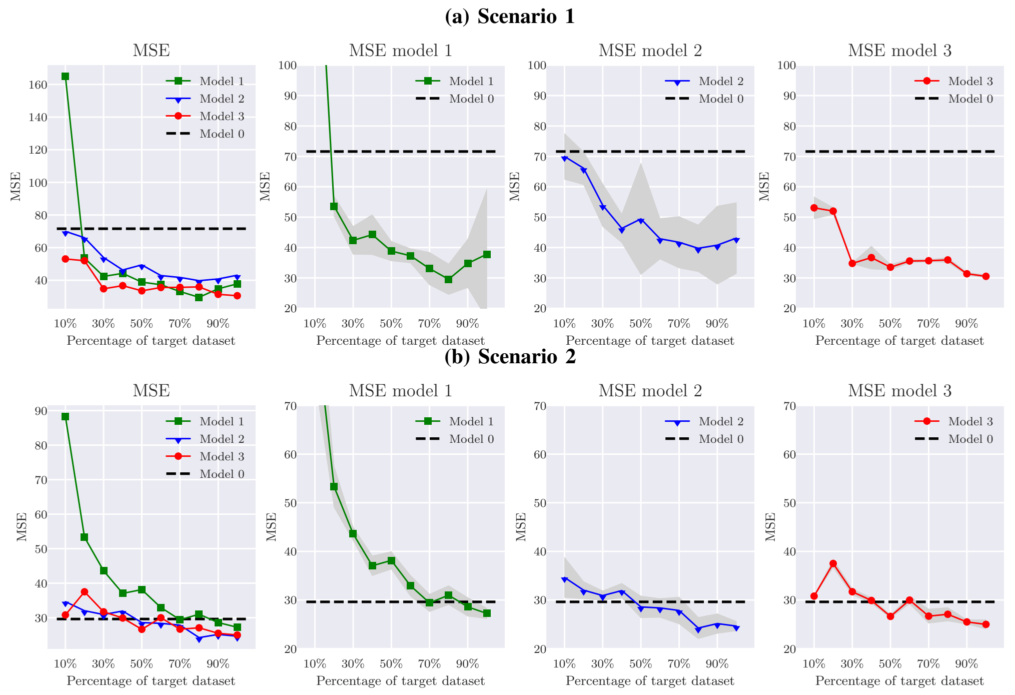

Fig. 5. $\overline{MSE}$ for $\{10\%, 20\%, \ldots, 100\%\}$ of target dataset: Scenario 1 (a) and Scenario 2 (b)

TABLE II
$\overline{MSE}$ (σ) ON 10 RUNS FOR $\{10\%, 20\%, \ldots, 100\%\}$ OF TARGET DATASET: SCENARIO 1 AND SCENARIO 2

	Scenario 1			**Scenario 2**		
	`Model_1`	`Model_2`	`Model_3` (TL)	`Model_1`	`Model_2`	`Model_3` (TL)
10%	165.05 (6.46)	69.96 (7.55)	53.06 (3.61)	88.28 (12.78)	34.63 (4.14)	30.80(0.06)
20%	53.60 (3.32)	66.07 (5.39)	51.99 (1.17)	53.31 (4.24)	32.02 (1.77)	37.50 (0.84)
30%	42.36 (4.59)	53.92 (7.04)	34.78 (0.17)	43.60 (1.39)	30.91 (0.97)	31.70 (0.62)
40%	44.24 (6.58)	46.30 (4.71)	36.69 (3.77)	37.05 (2.04)	31.93 (1.49)	29.90 (0.57)
50%	38.83 (3.22)	49.42 (18.32)	33.53 (0.88)	38.14 (1.81)	28.57 (2.24)	26.63 (0.31)
60%	37.29 (2.36)	42.87 (6.67)	35.56 (0.63)	32.94 (2.15)	28.37 (1.94)	29.98 (0.79)
70%	3.07 (5.36)	41.73 (8.46)	35.66 (0.37)	29.40 (1.80)	27.84 (2.76)	26.68 (1.46)
80%	29.56 (5.03)	39.75 (7.69)	35.92 (0.80)	31.00 (1.94)	24.23 (2.19)	21.06 (1.38)
90%	34.78 (7.95)	40.79 (12.84)	31.36 (0.51)	28.58 (1.91)	25.15 (2.02)	25.48 (0.45)
100%	37.81 (21.30)	43.11 (11.67)	30.53 (0.46)	27.23 (1.06)	24.63 (0.98)	24.99 (0.96)

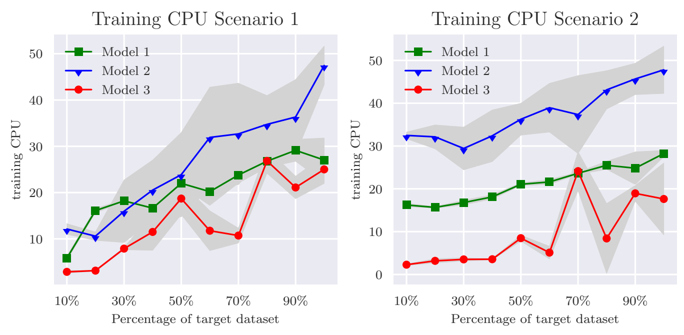

Fig. 6. Average training CPU for different percentages of target dataset for Scenario 1 and Scenario 2

are too important, and thus an update is necessary. For similar reasons, the model trained from scratch on the source data and the upcoming new data (the source and target datasets) fails too. Building from scratch a model with only the target data is possible, but requires engineering tests and training time. The TL is successful to update the VM CNN-based source model. It reaches a good accuracy with low training time, by transferring knowledge from the source domain to the target domain without having to learn from scratch.

When no new process conditions occur over time, the experimental results from Scenario 2 show that the transfer learning increases the stability of the prediction and saves the computational time. In addition, the TL provides a good level of accuracy even for small percentages of the target dataset.

This finding could be particularly valuable to be exploited for improving production sampling rates.

VII. Conclusions and perspectives

This paper focuses on updating a VM CNN-based model applied to predict the average material removal rate in a CMP process. Several strategies to update the VM CNN-based model have been compared within two scenarios. In the first scenario, the dynamic nature of the considered manufacturing process subject to drifts is explicitly considered. In the second scenario, no new process conditions occur over production cycles. The results showed that transfer learning succeeded to update the current model for both considered scenarios. In the first scenario, the TL overcomes the difference between the source domain (before sampling new wafers) and the target domain (after sampling new wafers), by transferring knowledge from the source domain to the target domain. The model with TL reaches a better accuracy with a lower training time than other investigated reference models. In the second scenario, where there is no noticeable difference between the source and target domains, the TL increases the stability of prediction and saves the computational time.

Following these promising results, we intend to investigate more deeply the capabilities of TL for the updatability (discussed in this paper) and adaptability (see [5]) features of VM models. In this paper, the sampling have been performed a priori and randomly. Physical metrology is time-consuming and expensive. Fostered by the derived findings, we also intend to explore the potential of TL to improve the performance of sampling decision systems, as another avenue for future research on the applications of VM.

References

[1] S. Kim, J. Jang, and C. O. Kim, "A run-to-run controller for a chemical mechanical planarization process using least squares generative adversarial networks," *Journal of Intelligent Manufacturing*, vol. 32, no. 8, pp. 2267–2280, 2021.

[2] P.-A. Dreyfus, F. Psarommatis, G. May, and D. Kiritsis, "Virtual metrology as an approach for product quality estimation in industry 4.0: a systematic review and integrative conceptual framework," *International Journal of Production Research*, vol. 0, no. 0, pp. 1–24, 2021.

[3] W.-M. Wu, F.-T. Cheng, and F.-W. Kong, "Dynamic-moving-window scheme for virtual-metrology model refreshing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 25, no. 2, pp. 238–246, 2012.

[4] L. L. T. Chan, X. Wu, J. Chen, L. Xie, and C. I. Chen, "Just-In-Time Modeling with Variable Shrinkage Based on Gaussian Processes for Semiconductor Manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 31, no. 3, pp. 335–342, 2018.

[5] R. Clain, V. Borodin, M. Juge, and A. Roussy, "Virtual metrology for semiconductor manufacturing: Focus on transfer learning," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 2021, pp. 1621–1626.

[6] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[7] P. D. C. P. Society, "PHM data challenge," https://www.phmsociety.org/events/conference/phm/16/data-challenge, 2016.

[8] J. Feng, X. Jia, F. Zhu, J. Moyne, J. Iskandar, and J. Lee, "An online virtual metrology model with sample selection for the tracking of dynamic manufacturing processes with slow drift," *IEEE Transactions on Semiconductor Manufacturing*, vol. 32, no. 4, pp. 574–582, 2019.

[9] H. Cai, J. Feng, F. Zhu, Q. Yang, X. Li, and J. Lee, "Adaptive virtual metrology method based on just-in-time reference and particle filter for semiconductor manufacturing," *Measurement*, vol. 168, p. 108338, 2021.

[10] X. Jia, Y. Di, J. Feng, Q. Yang, H. Dai, and J. Lee, "Adaptive virtual metrology for semiconductor chemical mechanical planarization process using GMDH-type polynomial neural networks," *Journal of Process Control*, vol. 62, pp. 44–54, feb 2018.

[11] T. E. Korabi, V. Borodin, M. Juge, and A. Roussy, "A hybrid feature selection approach for virtual metrology: Application to CMP process," in *2021 32nd Annual SEMI Advanced Semiconductor Manufacturing Conference (ASMC)*, 2021, pp. 1–5.