Proceedings of the 2023 Winter Simulation Conference
C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, eds.

TRANSFORMING DISCRETE EVENT MODELS TO MACHINE LEARNING MODELS

Hessam S. Sarjoughian

Arizona Center for Integrative
Modeling & Simulation
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, 85281, USA

Forouzan Fallah

Arizona Center for Integrative
Modeling & Simulation
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, 85281, USA

Seyyedamirhossein Saeidi

Arizona Center for Integrative
Modeling & Simulation
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ, 85281, USA

Edward J. Yellig

5000 W. Chandler Blvd

Intel Corporation
5000 W. Chandler Blvd
Chandler, AZ, 85226, USA

ABSTRACT

Discrete event simulation, formalized as deductive modeling, has been shown to be effective for studying dynamical systems. Development of models, however, is challenging when numerous interacting components are involved and should operate under different conditions. Machine Learning (ML) holds the promise to help reduce the effort needed to develop models. Toward this goal, a collection of ML algorithms, including Automatic Relevance Determination are used. Parallel Discrete Event System Specification (DEVS) models are developed for single-stage and cascade two-stage factories. Each model is simulated under different demand profiles. The simulated data sets are partitioned into subsets, each for one or more model components. The algorithms are applied to the data sets for generating ML models. The throughputs predicted by the ML models closely match those in the DEVS simulated data. This study contributes to modeling by demonstrating the potential benefits and complications of utilizing ML.

1 INTRODUCTION

Dynamical models are vital for understanding and operating complex systems, including smart manufacturing. These models must be able to simulate well-defined conditions, but development efforts and computing resources can vary greatly based on scale and complexity. A unifying concept for describing dynamical systems is to define them to have inputs, operations, and outputs. The two paradigms for model development are deductive and inductive. The former requires creating structure and behavior abstractions and the latter extracts some system structure and behavior from input and output data.

A deductive model is one where its outputs are computed given some known operations that act on some inputs. These kinds of models are generally based on first principles. Operations are commonly defined in terms of states, functions, and relations that transform some inputs into some outputs. Operations are formalized according to some form of cause-and-effect definition. Each model has its own structure and behavior. Considering the Discrete Event System Specification (DEVS) modeling method (Chow &

Zeigler, 1994), it is well-suited for describing and simulating the dynamics of discrete event systems. These models are generally defined as primitive and composite components.

In inductive modeling, the operations for some observed actual and/or simulated input and output regimes are to be identified. Such operations are expected to be applied to unobserved input regimes and generate acceptable output regimes. Machine Learning Algorithms (MLA) such as Linear Regression and Naïve Bayes (Bishop & Nasrabadi, 2006) broadly fall into the inductive modeling paradigm. The viability of MLA can vary significantly depending on the quantity and quality of a given system's input and output data as well as its behavior under different operating conditions. Considering discrete event systems, their data and output regimes are time-dependent and may behave deterministically or not.

Development of component-based models for new systems generally requires significant time and resources including the skill to create knowledge (Fujimoto et al., 2018). Changing these models is difficult or impractical in view of expert skill, time, and effort. As the scale and complexity of models grow, more powerful computational platforms are needed. Development and use of such parallel and distributed simulation platforms can be exceedingly time-consuming and expensive. Even having such models in hand, it may be practical to simulate them in a timely manner.

It can also be complicated to develop component models using ML. Models are difficult to generate as most algorithms are applicable to specific domains. Time does not serve a principal role, either explicitly or implicitly, in many MLA. Large data volumes and computing resources are generally needed for training and testing. A substantial amount of time may be needed to develop predictive models. Still, the development of these models is desirable since they may be extracted much faster compared to developing component-based models. MLA may be executed (simulated) much faster and benefit from specialized computing platforms, including GPU clusters when a vast amount of data is needed.

The above indicates deriving models using ML from component-based models stands to advance simulation studies under the assumption the necessary simulated data is available or otherwise can be obtained in a timely fashion. This research focuses on smart semiconductor manufacturing, a class of dynamical discrete event systems that have intricate structures and behaviors. Such factories with many kinds of machines rely on simulations to find efficient schedules for connected processes. Individual factories should enable the flexibility for producing computer chips through new kinds of connected factories. Simulations of these systems should be more flexible and highly efficient as the models' scale and complexity will inevitably grow.

This paper details an approach where two discrete-event models of re-entrant semiconductor manufacturing factories are used and transformed to ML models. Sections 2 and 3 provide a short background on parallel DEVS and supervised MLA and closely related works. Section 4 describes a method for devising simulation experiment scenarios, a scheme to choose and a process to synthesize the simulated data, and a selection of MLA using a re-entrant single-stage and a cascade two-stage semiconductor factory models. Section 5 details the experiments performed using the MLA and evaluating their prediction power. Section 6 concludes the paper and highlights future research.

2 BACKGROUND

Discrete-event models are generally created using first principles as deductive methods whereas MLA as inductive methods rely on second principles. There exist several approaches, frameworks, and tools for specifying and simulating discrete event models. Similarly, many MLA have been developed. As models, they represent some form of correlation among some given input and their processing as output data. The DEVS and MLA are chosen for this research. These are briefly described next.

2.1 Discrete Event Modeling

Considering component-based deductive modeling, the parallel atomic DEVS model specification has input, state, and output variables and time-based functions. The specification requires concise formulations for processing inputs, state transitions, and generating outputs. As primitive components, they can be

hierarchically composed to create a variety of composite models. Well-formed input and output provide concise semantics for individual components and their compositions. As a deductive modeling approach, it lends itself to concise understanding and causal reasoning. The input and output events may occur at arbitrary time instances. At an instance of time only input events may occur at time instances, and outputs are produced in non-uniform time intervals. These models can be simulated using a variety of frameworks and tools developed in popular programming languages and executable on single and multi-processor computing platforms supported with distributed technologies (e.g., web services). The execution of these models conforms to the parallel simulation protocol where operations in the models with their input and output interactions are entirely observable. Model structures (components, couplings, and hierarchy), model types, atomic and coupled behavioral complexities, and primitive/compound data types results in the durations of simulation executions varying from a few milliseconds to hours and days. This becomes important when many simulation experiments are needed for ML.

2.2 Machine Learning Algorithms

Numerous MLA (both supervised and unsupervised) are designed to extract models from available input and output data. The models make predictions that match unobserved data in varying degrees. As inductive models, such models define correlations between input and output variables. A simple linear model has a function with intercept and coefficient parameters. The function should satisfy some gradient descent measure such as the Root Mean Squared Error by finding the best values for the parameters for some given input and output data set. This model should minimize the difference between the true and predicted values for some outputs. This kind of model is holistic in contrast to the component-based modeling highlighted above. These algorithms are classified and evolved in past decades alongside the increasing amount of gathered data and huge advances in software and hardware. The K-Nearest Neighbors (KNN), Automatic Relevance Determination (ARD), and Decision Tree (DT) all are MLA that can be used for regression problems that predict a continuous target variable. A few aspects of these are noted next.

KNN operates by identifying the k closest neighbor points in the training set to the test point and predicting the target value based on the average or mode of their target values. The selection of k is a significant hyperparameter that affects the balance between overfitting and underfitting. KNN has the advantage of being able to capture nonlinear relationships between features and targets, making it an excellent choice for small datasets. ARD estimates the relevance of each feature in predicting the target variable. While Bayesian linear regression regularizes all features identically, it is not well-suited when only a few features are significant. In contrast, the regularization performed by this algorithm is highly adaptive since all weights are regularized differently (Thirion et al., 2011). It assigns a prior distribution to the regression coefficients, which promotes sparsity, meaning many of them will be set to zero. Relevant features will have non-zero coefficients, while irrelevant features will have coefficients that are close to zero. DT uses a process involving a sequence of binary selections through traversing an input data tree structure. It is a non-parametric algorithm that operates by recursively partitioning the feature space into smaller regions based on the features' values. Each partition corresponds to a node in the tree, with the leaf nodes representing the predicted values for the corresponding region. One benefit is the ability to handle nonlinear relationships between features and targets. It can also handle missing values and categorical features. However, DT can overfit the training data, particularly if the tree is too deep or the minimum number of samples per leaf is too small (Bishop & Nasrabadi, 2006). The scikit-learn library (Pedregosa et al., 2011) is used for all the implementations of the MLA in this paper.

3 RELATED WORKS

Research works can be viewed to focus on gaining simulation speed-up and reducing the effort to develop simulation models. Statistical regression modeling is proposed for the class of continuous systems that can be faithfully modeled as DEVS (Saadawi et al., 2016). This work proposes mapping the input, output, and event variables and transitions of the atomic models of a coupled model should be mapped to a collection

holistic set of variables and functions. A predictive model using regression may be incrementally built on the data. As a ML algorithm, it operates alongside the DEVS simulator. Therefore, the predictive model can be incrementally built using simulation runs. The amount of computation for creating the predictive model is expected to be less than what is needed for simulation. This research, like the work described in this paper, uses DEVS simulated data to into generate ML models. Considering semiconductor wafer factories, it is shown queuing models can be created from discrete-event models (Seok et al., 2020). Simulation speed-up is achieved by replacing high-fidelity model components with low-fidelity queuing models while achieving accuracy. Reinforcement learning for Digital Twin is proposed for the classic DEVS formalism models (David & Syriani, 2022). This work highlights shows that simulation models can adapt automatically to the changes observed in systems. Earlier research shows the use of recurrent neural networks to create the structure and behavior for finite memory DEVS models from some finite observed input and output event data (Choi & Kim, 2002). These works highlight the challenges of extracting models from observations obtained for connected semiconductor factories with re-entrant dynamics.

The prominence of semiconductor manufacturing has also attracted research In predictive simulation using ML. Numerous studies have been conducted in to address the challenges and opportunities presented by the data-driven digital landscape of semiconductor manufacturing (Kang et al., 2020). A new approach is proposed to predict and control cycle times in large-scale semiconductor manufacturing systems, by considering the interaction between workcentres through global information such as release quantities, Work-In-Process (WIP), cycle time targets, and machine capacities (Barhebwa-Mushamuka et al., 2023). The approach involves local scheduling decisions guided by production targets, which are determined by a constraint-based optimization for cycle time targets. Another study proposes a system to predict the throughput time of production processes in manufacturing using AutoML (regression) (Hiller et al., 2022). Regression analysis and DTs are used to develop predictive models. The case study demonstrated the application possibilities of throughput time predictions based on the provided systematization, and highlighted differences in data availability and prediction quality. The authors suggest that feature engineering is an area that requires more research to further promote the practical use of predictive models. In another work, the factors influencing cycle time and patterns are identified to create ML predictive models for estimating cycle time (Meidan et al., 2011).

A combination of clustering and regression analysis to Identify patterns in historical data and build models that could forecast cycle time has also been developed (Backus et al., 2006). Regression trees are used for obtaining predictions for lots in production based on completed lots that are similar. They can handle both categorical and continuous variables without scaling. Deep learning to predict jobs remaining time during manufacturing execution based on production big data has also been studied (Fang et al., 2020). The proposed method uses a stacked sparse autoencoder to learn representative features from high dimensional manufacturing big data to make accurate predictions. This study is applied in a large-scale job shop equipped with 44 machine tools producing 13 types of parts. ML techniques can also be used to identify factors affecting throughput in a semiconductor factory (Singgih, 2021). The study also contributed to the development of the Mini-Fab using Anylogic. It proposed a data collection scheme for the production control mechanism and classified the throughput into “good” and “bad” categories rather than quantitative predictions. Predicting throughput or cycle time at high accuracy is not considered essential for purposes such as identifying input/output patterns.

4 TRANSFORMING DEVS MODELS TO ML MODELS

This research shows an approach for extracting models using MLA applied to the data obtained from sets of discrete event simulation experiments. The approach can be viewed as having four steps. First, DEVS are developed and verified for correctness. Second, a set of experiments are devised, simulated, and validated. Third, the simulation data sets are processed and structured in terms of input (features) and output data (labels) sets to be used with MLA. Third, a collection of MLA is selected and employed with the input variables to predict output variables. This step involves choosing training and test data sets. Fourth, the

strengths of MLA are evaluated in terms of quantitative metrics and DEVS and ML modelers' intuitions for the application domain of interest.

Two DEVS models (single-stage and two-stage cascade semiconductor factories) are developed and simulated using the DEVS-Suite simulator. ARD, Bayesian, DT, KNN, LassoLars, Passive Aggressive Regression, Support Vector Regression (SVR), and TheilSen are selected. The simulated Mini-Fab input data sets are partitioned according to their scales. The training and test data are randomly and deterministically selected from the portioned data sets. The random choice of training and test data is for evaluating the accuracy of output predictions. The deterministic choices of training and test data is aimed at evaluating the MLA learning ability.

4.1 Simulation Models

A DEVS single-stage factory and a larger and more complex two-stage cascade factory are developed. The single-stage model is developed based on an existing benchmark (Kempf, 1994). A simplified component diagram for this model is shown in Figure 1. This factory consists of *Diffusion*, *Implantation*, and *Lithography* parts having machines A, …, E. Each part has a coordinator that dispatches wafers to one of the machines. It has 11 atomic and 4 coupled models and the Implantation model has 30 couplings. Four transducer models are used to measure and collect input, output, and state information at every simulation execution step. These transducers have no influence in the operations of the components and their couplings. A two-stage cascade model is conceptualized and created based on the single-stage model. The Lithography machine of the single-stage model is changed for coupling Stage-1 and Stage-2 (see Figure 1). This cascade model has 29 atomic and 9 coupled components. Seven transducer models are used to measure and collect input, output, and state measurements at every simulation execution step. This cascade model will be used to generate simulation scenarios that exhibit higher structure and behavior complexities. The factory models can receive three kinds of wafers called Pa, Pb, and Tw. The number for Pa and Pb can be {0,2,3,6,9,12,18,24,27,30,36,45,48,54,60,72,81,90}. The number for Tw can be {0,1,3,6,9,12,15,18,27}.

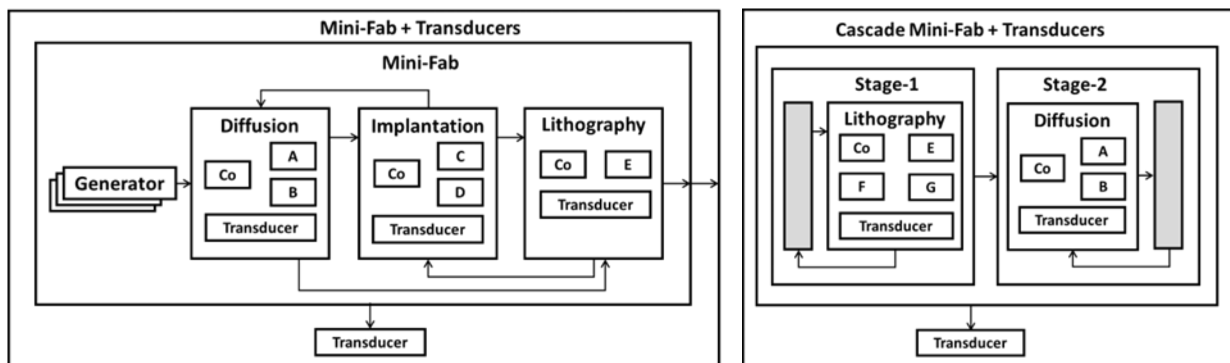

Figure 1: Single-stage and two-stage cascade semiconductor factory models.

4.2 Simulation Data Sets

A set of experiments are conducted for the Mini-Fab shown in Figure 1. Three types of wafer lots, namely product Pa, product Pb, and test wafer lot TW, undergo processing through six steps. The lots need to form a batch with a size of three prior to being processed. The possible number of experiments defined in terms of (Pa, Pb, Tw) tuples is 218,700 out of which 5,000 are considered suitable for this study, since the rules for batching are that only one TW lot can be included in the batch at most while Pa and Pb can be mixed in step 1, they cannot be mixed in step 5. The timing and other properties of these models can vary. For this research, 51 simulation experiments are devised and executed. The simulated data for the configured

experiments are divided into three groups. The first has 24 configurations (called *Group-1*) with the total number of wafers ranging from 6, 12, 18, 33, 36, to 54. The second has 16 configurations (called *Group-2*) with the total wafers being 54 and 72. The third has 11 configurations (called *Group-3*) with the total number of wafers being 90, 108, and 132. A total of 70 variables are measured via five machines, three coordinators, three generators, and four transducers for each of the 51 simulations. The logical durations of these experiments range from 22799 to 65297 minutes. However, simulating and collecting all data from these simulations consumes several hours when the total number of Pa, Pb, and Tw is 132 variables that are measured for Machines A, B, and E as well as their transducers for the Diffusion and Lithography components shown in Table 1. The variables for Machines C and D and its transducer are the same as for Machines A and B and their transducers. The collection of 51 simulations is considered a complete data set for use in MLA. Another set of experiments is conducted for the two-stage cascade factory. A total of 142 measurements are collected.

The number of execution steps varies among simulation runs. For example, an experiment for (6,9,3) configuration completes after 397 steps. Experiment for (36,18,0) and (0-72-0) configurations complete after 1,229 and 1,665 steps, respectively. This includes simulation steps for processing multiple input events and internal state transitions in zero-time logical simulation steps amongst all atomic models. The number of input and output events varies among the diffusion, implantation, and lithography components. The differences in the logic and timing of the operations in the machines can generate varied behaviors. The data sets are collected from the experiments where all jobs received are completely processed for the single-stage and two-stage models. The data collected at the end of the 51 simulation runs are used for the study of MLA.

5 EXPERIMENTS AND MODEL TRANSFORMATION EVALUATION

The use of the Mini-Fab DEVS model for deriving ML models is evaluated by designing, developing, and evaluating the experiments highlighted in Section 4. The collected experimental data for a single-stage Mini-Fab are divided into 18 scenarios corresponding to the modular structure shown in Figure 1. The scenarios for MLA have Diffusion D, Implantation I and Lithograph L components with their corresponding transducers T_D, T_I, T_L and transducer T_{MF} for the Mini-Fab (see Table 2). These simulated data sets are named *Sim-Data-1*, …, *Sim-Data-18*. For example, nine MLA are applied to Sim-Data-10 with 28 features (i.e., the data collected from the Implantation, Diffusion, and Lithography components). The acronyms SD stand for the selection of machine and transducer components and FS for the number of features used in each scenario.

Table 1: Machine, coordinator, and transducer components.

A, B, Transducer (24 variables)	**E, Transducer (16 variables)**	**Mini-Fab (6 variables)**
• A & B *Load time* • A & B *Processing time* 1 & 5 • A & B *Transport time* • A & B *Unload time* • A & B *Step* 1 & 5 • A & B *Throughput* • $CoAB$ 1 & 5 *Throughput* • A & B *Turnaround time* • $CoAB$ 1 & 5 *Turnaround time*	• *Load time* • *Processing time* 3 & 6 • *Transport time* • *Unload time* • *Setup time* • E *Step* 3 & 6 • E *Throughput* • CoE 3 & 6 *Throughput* • E *Turnaround time* • CoE 3 & 6 *Turnaround time*	• *Throughput time* • *Turnaround time* $CoAB$: Coordinator for Implantation component CoE: Coordinator for Lithography component

5.1 Single-Stage Mini-Fab

The predictions of a set of ML models using the common 80/20 percentages for training and test are compared with actual simulation output. The 70/30, 60/40, and 50/50 ratios are also used for evaluating the nine MLA. The results are similar to those obtained for the 80/20 ratio.

Table 2: Simulation experiments scenarios.

SD	Components	FS	SD	Components	FS	SD	Components	FS
1	D	14	7	$D+I$	13	13	$I+T_I+L+T_L$	35
2	$D+T_D$	22	8	$D+L$	36	14	$D+T_D+I+T_I+L+T_L$	57
3	I	14	9	$I+L$	22	15	$D+T_D+I+T_I+L+T_L+T_{MF}$	59
4	$I+T_I$	22	10	$D+I+L$	28	16	SD#15, excludes TH for D, I, L	51
5	L	44	11	$D+T_D+I+T_I$	35	17	SD#15, excludes TA for D, I, L	50
6	$L+T_L$	8	12	$D+T_D+L+T_L$	22	18	SD#16, excludes TA for D, I, L	39

Table 3: Measurements for the ARD, DT, and KNN single-stage and two-stage cascade models.

	Single-Stage																		Two-Stage			
SD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	1	2	3	4
MAE	4.04E-05	2.43E-05	6.37E-05	3.77E-05	2.43E-05	3.07E-05	3.99E-06	2.44E-05	2.04E-05	5.43E-05	1.69E-05	2.75E-05	1.64E-05	3.97E-06	3.97E-06	2.21E-05	3.99E-06	2.44E-05	5.56E-05	6.67E-05	5.83E-05	2.33E-05
R^2	0.995176	0.998898	0.991162	0.996039	0.998898	0.998449	0.999977	0.998895	0.999322	0.992157	0.999546	0.998505	0.999519	0.999978	0.999978	0.999241	0.999977	0.998895	0.989065	0.986878	0.988609	0.997996
MSE	5.E-09	1E-09	8.00E-09	3.73E-09	1.00E-09	1.46E-09	2.14E-11	1.00E-09	6.39E-10	7.00E-09	4.28E-10	1.41E-09	4.54E-10	2.12E-11	2.12E-11	7.15E-10	2.14E-11	1.00E-09	6.00E-09	7.00E-09	6.00E-09	1.02E-09
Model	K = 1	K = 1	K = 1	D = 32	K = 1	D = 10	ARD	K = 1	D = 34	K = 1	D = 35	D = 14	D = 13	ARD	ARD	D = 37	ARD	K = 1	K = 6	K = 1	K = 6	ARD

The algorithms in Section 3 are applied to the simulated data scenarios (see Table 2). The throughput predictions by ARD and DT are closest to the simulated data. The Mean Absolute Error (MAE), Mean squared error (MSE), and R-squared Error (R^2) measurements have small differences (see Table 3). The ARD model generates the best throughput compared with the DEVS simulation. It has the smallest number of features for scenarios 7, 14, 15, and 17. The depth of the DT model changes while the number of the neighbors for KNN does not.

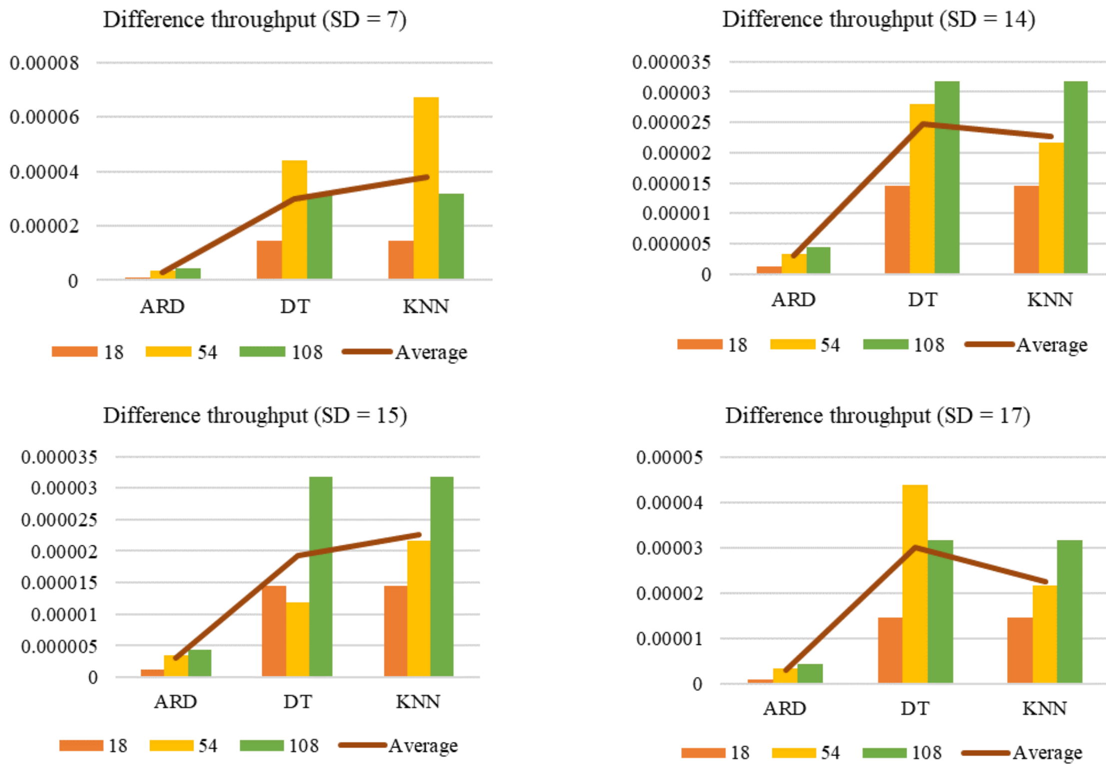

Figure 2: Comparison of DEVS model throughputs for the ARD, DT, and KNN models.

Learning: Generating ML models involves using algorithms and statistical techniques to learn from data. One key aspect of this process is learning during the training phase. To understand and quantify the learning of ARD, a set of experiments are devised for selected demand profiles and training data sets. Four sets of experiment configurations are considered. The test demand profiles for the simulations are shown in Table 4. As shown in Figure 3, the number of test data changes to account for larger training data sets. First, a pair of experiments called A-1 and A-2 using 14 data sets is devised for 80 percent for training and 20% for testing. The range of Pa, Pb, and Tw range from 6 to 18. The test data for A-2 has three unseen simulations with Pa, Pb, and Tw totaling to 36 and 54. Second, a pair of experiments called B-1 and B-2 are also devised using 30 data sets with 80/20 training and test data ratio. The range of Pa, Pb, and Tw range from 6 to 54. The test data for B-2 is the same A-2. The results in Figure 3 show the MLA performs better by increasing the training data from 11 to 24. The MSE is reduced by 98.5% and R^2 is increased by11.2%. Third, a pair of experiments called C-1 and C-2 is devised similar to B-1 and B-2, but with the test data sets for Pa, Pb, and Tw totaling to 90 and 132. Forth, a pair of experiments called D-1 and D-2 are devised using 51 simulations. The MSE is reduced by 89.9% and R^2 is increased by 0.7% for C-2 and D-2. It should be noted that the number of simulations is significantly smaller for D-1 and D-2 compared to C-1 and C-2 and even more compared to A-1 and A-2 (see Section 4.2). Despite much smaller number of simulations, the ARD model shows it can still learn. Thus, given the exponential growth in the number of simulation experiments, it is expected for ARD model learning to improve in accordance with the MAE, MSE, and R^2 measurements. These ML models demand increasingly need more computation and longer time for simulations, data collection, and evaluations.

Table 4: Pa, Pb, and Tw configurations for ARD model.

A1			B1			A2 & B2			C1			D1			C2 & D2		
Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw	Pa	Pb	Tw
9	9	0	27	18	9	12	18	6	18	27	9	54	0	18	30	45	15
12	6	0	27	27	0	54	0	0	18	36	0	54	30	15	45	45	0
18	0	0	24	36	12	36	18	0	54	0	0	90	90	27	60	60	12

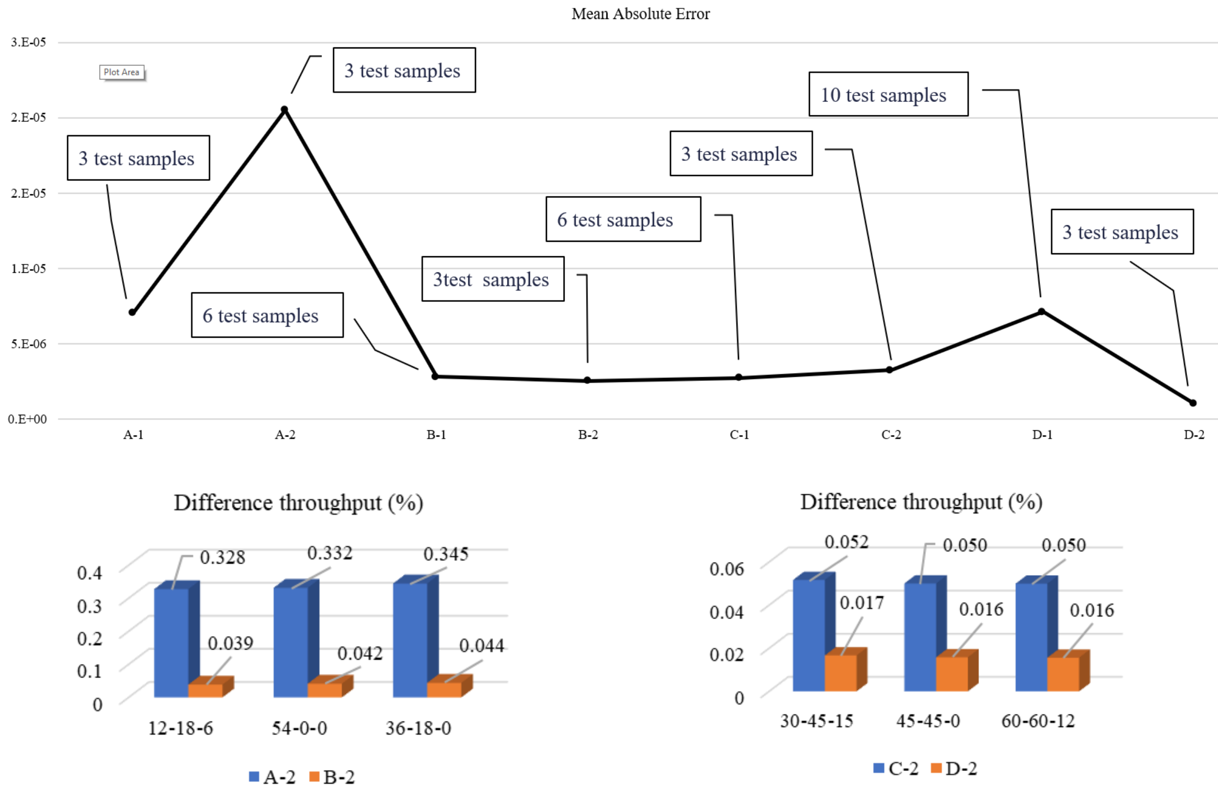

Figure 3: ARD model throughput predictions improve due to an increase in the number of simulation scenarios.

5.2 Two-Stage Mini-Fab

To further examine the use of DEVS to ML models, the single-stage simulation experimentation is carried out for the two-stage cascade factory model. The number of experiments for the two-stage factory is 26 out of which 22 are used for training and four for testing. The total numbers of Pa, Pb, and Tw for these simulations range from 6 to 72. As in the single-stage, all simulation executions process all Pa, Pb, and Tw wafers. The number of wafers in test scenarios has 18, 54, and 72. The throughput predictions for the ARD, DT, and KNN are shown in Figure 4. The ARD performs better than DT and KNN models. The increases in the throughput differences in ARD, DT, and KNN from 18 to 54 is expected. However, the decreases

from 54 to 72 are not. This may be due to far fewer simulation experiments and having 29 components compared with the 11 components for the single-stage model.

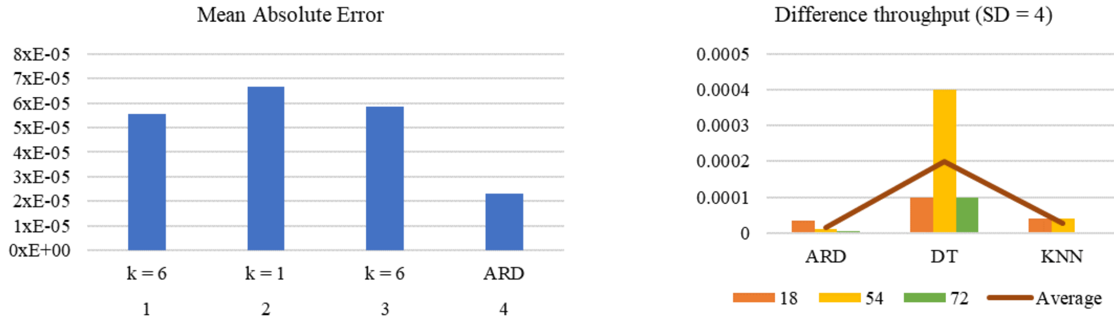

Figure 4: Comparison of DEVS model throughputs for the ARD, DT, and KNN models and MAEs for simulation scenarios.

5.3 Discussion

The approach for transforming DEVS models into ML models is shown to be suitable for the single-stage and two-stage models. The data collected from simulations have irregular time durations. The ARD, DT, and KNN algorithms, however, cannot be event-based input and output and state variables. Therefore, the ML models do not predict throughputs in time steps for the Mini-Fab models. In other words, it is not possible for the ML models to predict throughput chronologically. In contrast, the simulation models are equipped to predict changes in the throughputs of individual machines or combinations of the machines in the diffusion, implantation, and lithography parts of the factories. Predicting the dynamics of models may be achieved with algorithms that can use time-based data sets for training and testing. This is a critical when actual systems or simulation models do not exist. Approaches such as Temporal Convolutional Networks (Lea et al., 2016) are aimed at graph representation learning using time-series, not discrete-event data. Discrete event systems are naturally reactive and do not operate on regulated time steps.

In transforming DEVS models into ML models, some of the simulated data are small. This can lead to very small MAE or MSE values. This may result in ML model throughput predictions that are inaccurate or difficult to understand and explain. Therefore, it is important to consider the scale of the simulated data sets when developing ML models. The predicted output should be carefully examined. While scaling or normalization techniques are available, using them may alter the actual values and potentially change the nature of the data for the MLA. Additionally, one assumption made in this context is that the dataset is continuous, despite the fact that the models are discrete-event with added randomness in timing and non-deterministic logic. Also, the study assumes that the data distribution and properties remain constant over time, which may not always be the case in real-world scenarios. These limitations suggest opportunities for future research to develop more robust models that can handle variations in data properties and distribution or small values and provide accurate predictions with limited training data.

6 CONCLUSION AND FUTURE RESEARCH

In this paper, transforming DEVS models into ML models is investigated through a series of experiments for two semiconductor factory models. A test bed is developed in the DEVS-Suite Simulator for a one-stage Mini-Fab model, followed by the development of a two-stage Mini-Fab model. A scheme is also developed to manage and process large amounts of data. The use of different ML regression models with various feature selections and knowledge of simulation models has produced accurate throughput

predictions, offering a basis for future connected factories that require accurate and fast predictions. This study shows ML models have the ability to be used to predict the key characteristics of discrete-event systems. Future research needs to develop factory models under different structural topologies, operating conditions, and interaction modalities. Robust ML models should handle variations in data properties and distribution. It is also important to explore other machine learning approaches, measure speed-up gains in machine learning, and combine DEVS and ML models enabling verifiable and accelerated simulations.

ACKNOWLEDGMENTS

This research is funded by Intel Corporation, Chandler, Arizona, USA.

REFERENCES

Backus, P., Janakiram, M., Mowzoon, S., Runger, G. C., & Bhargava, A. (2006). Factory cycle-time prediction with a data-mining approach. *IEEE Transactions on Semiconductor Manufacturing*, *19*(2), 252–258. https://doi.org/10.1109/TSM.2006.873400

Barhebwa-Mushamuka, F., Dauzère-Pérès, S., & Yugma, C. (2023). A global scheduling approach for cycle time control in complex manufacturing systems. *International Journal of Production Research*, *61*(2), 559–579. https://doi.org/10.1080/00207543.2021.2010828

Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and machine learning* (Vol. 4, Issue 4). Springer.

Choi, S. J., & Kim, T. G. (2002). Identification of Discrete Event Systems Using the Compound Recurrent Neural Network: Extracting DEVS from Trained Network. Transaction of Modeling and Simulation International. 78(2), 90-104.

Chow, A. C. H., & Zeigler, B. P. (1994). Parallel DEVS: a parallel, hierarchical, modular modeling formalism. In J. D. T. S. M. D. A. Sadowski A. F. Seila (Ed.), *Proceedings of Winter Simulation Conference* (pp. 716–722). https://doi.org/10.1109/WSC.1994.717419

David, I., & Syriani, E. (2022). Devs model construction as a reinforcement learning problem. Annual Modeling and Simulation Conference (ANNSIM), San Diego, CA, USA.

Fang, W., Guo, Y., Liao, W., Ramani, K., & Huang, S. (2020). Big data driven jobs remaining time prediction in discrete manufacturing system: a deep learning-based approach. *International Journal of Production Research*, *58*(9), 2751–2766. https://doi.org/10.1080/00207543.2019.1602744

Fujimoto, R., Barjis, J., Blasch, E., Cai, W., Jin, D., Lee, S., & Son, Y.-J. (2018). Dynamic data driven application systems: research challenges and opportunities. In B. Johansson, S. Jain, O. Rose, M. Rabe, A. Skoogh, N. Mustafee, *Proceedings of Winter Simulation Conference (pp. 664-678).* https://doi.org/10.1109/WSC.2018.8632379

Hiller, T., Deipenwisch, L., & Nyhuis, P. (2022). Systemising Data-driven Methods for Predicting Throughput Time within Production Planning & Control. *IEEE International Conference on Industrial Engineering and Engineering Management*, *2022-December*, 716–721. https://doi.org/10.1109/IEEM55944.2022.9989885

Kang, Z., Catal, C., & Tekinerdogan, B. (2020). Machine learning applications in production lines: A systematic literature review. *Computers & Industrial Engineering*, *149*, 106773.

Kempf, K. (1994). Detailed description of a two-product five-machine six step re-entrant semiconductor manufacturing system. *Intel Corporation, Technology & Manufacturing Group*.

Lea, C., Vidal, R., Reiter, A., & Hager, G. D. (2016). Temporal convolutional networks: A unified approach to action segmentation. *Computer Vision–ECCV 2016 Workshops*: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14, 47–54.

Meidan, Y., Lerner, B., Rabinowitz, G., & Hassoun, M. (2011). Cycle-time key factor identification and prediction in semiconductor manufacturing using machine learning and data mining. *IEEE*

Transactions on Semiconductor Manufacturing, *24*(2), 237–248. https://doi.org/10.1109/TSM.2011.2118775

Saadawi, H., Wainer, G., Pliego, G., & Paul, S. (2016). DEVS execution acceleration with machine learning; DEVS execution acceleration with machine learning. In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*. https://doi.org/10.23919/TMS.2016.7918816

Seok, M. G. I., Chan, C. W., Cai, W., Sarjoughian, H. S., & Park, D. (2020). Runtime abstraction-level conversion of discrete-event wafer-fabrication models for simulation acceleration. *SIGSIM-PADS 2020 - Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 83–92. https://doi.org/10.1145/3384441.3395982

Singgih, I. K. (2021). Production flow analysis in a semiconductor fab using machine learning techniques. *Processes*, *9*(3), 1–18. https://doi.org/10.3390/pr9030407

Thirion, B., Michel, V., Eger, E., & Keribin, C. (2011). Multiclass sparse Bayesian regression for fMRI-based prediction. *International Journal of Biomedical Imaging*, *2011*. https://doi.org/10.1155/2011/350838

AUTHOR BIOGRAPHIES

HESSAM S. SARJOUGHIAN is an Associate Professor of Computer Science and Computer Engineering in the School of Computing and Augmented Intelligence (SCAI) at Arizona State University (ASU), Tempe, Arizona. He is the co-dirctor of the Arizona Center for Integrative Modeling and Simulation (https://acims.asu.edu). He can be reached at hessam.sarjoughian@asu.edu.

FOROUZAN FALLAH is a Ph.D. student in the Computer Science program in the School of Computing and Augmented Intelligence (SCAI) at Arizona State University, Tempe, AZ, USA. She can be reached at ffallah@asu.edu**.**

SEYYEDAMIRHOSSEIN SAEIDI is a Ph.D. student in the Computer Science program in the School of Computing and Augmented Intelligence (SCAI) at Arizona State University, Tempe, AZ, USA. He can be reached at ssaeidi1@asu.edu.

EDWARD J. YELLIG is the director of Operational Decisions Support Technology at Intel Corporation. He has been with Intel for 26 years and has a Ph.D. in Operations Research with the emphasis in discrete event modeling of large scale systems. His focus has been in developing fab models for determining capital requirements and is also responsible for the real-time digital twin tactical models. He can be reached at Edward.J.Yellig@Intel.com.