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

DEEP LEARNING ENABLING DIGITAL TWIN APPLICATIONS IN PRODUCTION SCHEDULING: CASE OF FLEXIBLE JOB SHOP MANUFACTURING ENVIRONMENT

Amir Ghasemi

Amsterdam School of International Business
Amsterdam University of Applied Sciences
Fraijlemaborg 133
Amsterdam, 1102 CV, NETHERLANDS

Yavar Taheri Yeganeh
Andrea Matta

Department of Mechanical Engineering
Politecnico di Milano
Milan, 20156, ITALY

Kamil Erkan Kabak

Dept. of Industrial Engineering
Izmir University of Economics
Izmir, 35330, TURKEY

Cathal Heavey

CONFIRM Research Centre
School of Engineering
University of Limerick
Limerick, V94 T9PX, IRELAND

ABSTRACT

Digital twin-based Production Scheduling (DTPS) is a process in which a digital model replicates a manufacturing system, known as a "Digital Twin (DT)". DT is essentially a virtual representation of physical equipment and processes that are connected to the physical environment using an online data-sharing infrastructure within the Manufacturing Execution System (MES). In the case of reactive scheduling, DT is used to detect fluctuations in the scheduling plan and execute rescheduling plans. In proactive scheduling, it is used to simulate different production scenarios and optimize future states of production operations. Replicating detailed simulation models in most PS cases is highly computationally intensive, which negates against the main goal of DT (online decision making). Thus, this research aims to examine the possibility of using data-driven models within the DT of a Flexible Job Shop (FJS) production environment aiming to provide online estimations of PS metrics enabling DT-based reactive/proactive scheduling.

1 INTRODUCTION

Digital Twin (DT), as one of the key technologies derived from Industry 4.0, refers to a virtual representation of a physical system or object, such as a machine, a production line, or a factory. This virtual model is created using real-time data obtained from sensors, Internet of Things (IoT), and other data sources to simulate the behavior of the physical system (Liu et al. 2021). DTs provide manufacturers with real-time monitoring and analysis of the performance of their physical assets, allowing them to optimize performance and identify areas for improvement.

Production Scheduling (PS) is a critical module within manufacturing systems planning that involves creating a detailed plan for production processes. The production schedule usually outlines the sequence of operations, the timing of each operation, and the resources required for each operation. Today, in the globalized and competitive market, there is no doubt about the crucial role of PS modules in manufacturing systems that allow manufacturers to utilize their resources efficiently and effectively to meet production goals. Within this context, reactive and proactive scheduling are two distinct approaches for PS in manufacturing. Reactive scheduling is characterized mainly by the need to respond to unforeseen or unexpected events that

disrupt the planned production schedule. Such disruptions may be caused by machine failures, supply chain delays, or other unforeseen events that require immediate response. In this approach, a decision maker (a human scheduler or a robot) is responsible for making quick decisions and communicating changes to the production unit (Takeda Berger et al. 2019). Proactive scheduling, on the other hand, involves anticipating potential disruptions and taking preventive measures to minimize their impact on the production schedule. This approach involves using tools to analyze data, predict future states of the production environment, and generate optimized production schedules based on these various scenarios. Proactive scheduling may consider factors such as demand forecasts, expected machine downtime for maintenance, and potential delays in each production step (e.g., processing time fluctuations) (Zhang et al. 2022). In both discussed PS strategies, the existence of a control system with data sharing capabilities is the key required infrastructure. In modern manufacturing systems, such as semiconductor manufacturing, this infrastructure is usually provided by the Manufacturing Execution System (MES). MES typically provides functionality for real-time data collection, PS, work order management, quality control, material management, and maintenance management. MES systems capture data from various sources, such as machines, sensors, and manual input from operators on the shop floor. This data is then analyzed and used to monitor production performance, identify bottlenecks, and make real-time decisions to optimize production production (Jaskó et al. 2020). As discussed above, DT is one of the tools proposed within the context of Industry 4.0 to provide real-time decisions regarding different aspects of a production unit, such as PS.

Accordingly, Digital twin-based Production Scheduling (DTPS) refers to generating scheduling plans based on fully or partially online simulation experiments using reactive, proactive, and/or hybrid PS strategies. These experiments (depending on the decision-making procedure) include online simulation models of the production unit and predictive simulation experiments examining future PS states. This is facilitated through the online data sharing infrastructure provided by the MES.

In recent years, there has been a growing interest in DTPS within the scientific community, particularly in the context of Industry 4.0 (Negri et al. 2020). In a recent study, Li et al. (2023) developed a reactive PS strategy using a DT. Their proposed architecture consists of a) Data Acquisition / Handheld terminals: to capture and transmit a physical job-shop production environment; b) Job-Shop Service System: to provide optimized reactive scheduling plans; c) Virtual Job Shop: to simulate the physical environment in an online manner. Within this architecture, the moment Job Shop Service System detects a PS anomaly (e.g., due to state, load, and processing time failures), it starts to develop rescheduling plans using a Grey Wolf Optimizer integrated with the online simulation model. Within the same research domain, Zhang et al. (2022) proposed a proactive PS methodology to be used within the DT of a job shop production environments. In their proposed methodology, local adjustment and right-shift strategies were used to minimize the total makespan of upcoming operations. Their research mainly proposed a theoretical framework for proactive DTPS, however, they did not discuss the required data sharing infrastructure (MES connection) required to execute such methodologies.

In other research, Negri et al. (2021) proposed a framework for real-time scheduling in a flow shop environment that integrates a decision tree module, a Genetic Algorithm (GA)-based optimization module, and a DT. The framework uses real-time data from accelerator sensors to compute the failure probability of the equipment and to simulate the performance of different production schedules. Their results demonstrated the viability of the framework and its applicability to flow shops and suggested that it could be adapted to other types of production system. Their article also discussed the importance of considering the standard deviation of performances in addition to makespan when selecting the best schedules. Villalonga et al. (2021) designed and implemented a framework for automated decision-making in cyber-physical production processes using DT-based architectures. Their proposed framework combined a fuzzy inference system and a scheduling procedure to provide efficient decision-making at both the local and global levels of a production system. Their framework integrates local DT into a global representation and uses real-time data collection and analysis for dynamic decision making. Their results showed that the proposed architecture outperforms the available solutions on the market in terms of flexibility and automated decision making.

However, the computing constraints for re-scheduling was their main declared limitation. On the one hand, one of the key features of DTPS is the ability to make online scheduling decisions using simulation and optimization experiments, however, the execution of a large number of simulation experiments to measure scheduling metrics in complex manufacturing systems is highly computationally intensive (Ghasemi et al. 2021).

ML models are an alternative to estimate PS metrics within complex manufacturing systems (Usuga Cadavid et al. 2020). In general, using ML models to estimate PS metrics consists of training, testing, tuning, and estimating steps. In more recent research, Li et al. (2023) proposed a Deep Reinforcement Learning (DRL) agent to minimize total tardiness in a Permutation Flow Shop Scheduling Problem (PMSP) with family setups. The proposed agent uses novel variable-length representations for state and action, which enables the agent to calculate a comprehensive priority for each job and select the job based on these priorities every time a machine is idle. The agent uses a Recurrent Neural Networks (RNN), specifically Gated Recurrent Unit (GRU), to approximate the agent policy, which captures the sequential relationships between jobs and handles variable-length sequences. They designed a two-stage training strategy to train the agent under a sparse reward function. The main limitations of their research are lack of dynamic and uncertain constraints such as new job insertions and machine breakdowns within their studied PS environment.

Jeong et al. (2023) addressed the scheduling problem of a wet clean station in semiconductor manufacturing where information on internal operations and settings is not available. To overcome this, their study proposed ML-based dispatch methods to find the best combination of recipes with the objective of minimizing the makespan. ML models were driven using data obtained from the track in and out times of each wafer lot, and a method was suggested to convert a sequence of recipes into input data in a matrix structure. The proposed ML models predicted the times that wafer lots spend in the wet clean station, depending on the combination of recipes sequentially processed in the station. Through a series of computational tests, the proposed models were evaluated and confirmed to be applicable to an actual fab, improving the productivity of the wet clean station. The study demonstrated the possibility of developing ML models for scheduling lots in an environment where there is no progress history inside the facility. However, the study was limited to only 10 product types. In earlier work on data-driven methods, Stricker et al. (2018) presented a new data-driven DRL algorithm designed to improve the operational performance of complex semiconductor production systems. In this example, the model assumes that the system state information is fully observable in real time, which is not always be the case in practice.

In summary, the computational intensity of simulation experiments prevents DT from being used as a tool for PS in complex manufacturing systems. To alleviate this issue, trained ML models can assist the DT in estimating the required scheduling metrics without executing a large number of simulation experiments. However, the application of ML models within the literature is mainly limited to selecting dispatching rules. Moreover, to simplify the ML application domain, within the literature, several critical aspects of real-time dynamic scheduling of manufacturing systems are ignored. Thus, in this research, the following content is provided:

- A simulation model that simulates the behaviors of the photolithography workstations in a semiconductor front-end fab as a Flexible Job Shop (FJS) environment. The considered FJS is considered at time point t with stochastic initial machines blockage times, stochastic sequence dependent setup times, and stochastic processing times.
- An ML-based solution (i.e., metamodel) for predicting makespan is developed. The methodology utilizes deep learning methods to capture the complexities of the FJS environment. Specifically, the architecture of the DL models are based on RNNs and attention mechanism.

The remainder of this article is organized as follows. Section 2 describes the considered manufacturing system's environment. Section 3 details parameters of the simulation model and the architecture of the generated data sets for ML. Section 4 details the configuration of the DL models used within this research.

Section 5 provides the experiments results after training and testing the proposed ML models. Section 6 discusses the findings, while Section 7 concludes the article and presents potential avenues for future research.

2 PROBLEM DESCRIPTION

Figure 1 shows the general research framework considered in this article. In this study, we examine a fully automated manufacturing cell, which bears resemblance to a photolithography environment in semiconductor manufacturing systems. In semiconductor manufacturing fabs, photolithography tools are fully automated sequences of several different toolsets in tendon. With high financial stakes, they are more than often pushed into high levels of utilizations for throughput optimization. This dictates most of the time their operation sequences to follow some preferable patterns with reasonable dispatching algorithms. In other words, photolithography tools are the least likely toolsets in semiconductor fabs to operate as flexible cells. Acordinglly, the flexiblity in the production cell is limited in this research. Typically, such manufacturing cells incorporate a cell controller, which integrates a combination of software and hardware components, including sensors, actuators, Programmable Logic Controllers (PLCs), and Supervisory Control and Data Acquisition (SCADA) systems, to administer manufacturing processes. The cell controller establishes a communication link with the MES, receiving directives and data on the type, quantity, and timing of the products to produce. The cell controller employs this information to manage the equipment in the manufacturing cell and generate the required products. Furthermore, the cell controller acquires production process data, such as machine status, cycle times, and production counts, which are transmitted to the MES for monitoring and analysis. This feedback mechanism enables the MES to optimize production and ensure that the manufacturing process is functioning efficiently.

Within the MES system, the Transaction Server, Database, and Web Server are integral components. The Transaction Server acts as a mediator between the MES system and other manufacturing systems, like the Cell Controller, and manages the transactions between them. The database, on the other hand, is the central storage location for all manufacturing data within the MES system. This includes essential information such as production schedules, inventory levels, equipment status, quality control data, and other metrics. The database plays a critical role in ensuring data accuracy and consistency, while also providing a reporting and analysis platform for users to gain insights from the data. Finally, the Web Server serves as the interface for users to interact with the MES system through a web browser. Tasks such as scheduling production runs, monitoring equipment status, and reviewing production data can be performed through the user interface provided by the Web Server to the Web Client.

In this study, the objective is to develop an ML model capable of accurately estimating PS metrics for a manufacturing cell. To achieve this goal, it is necessary to gather data related to jobs, scheduling plans, machines, queues, and actual performance metrics obtained for each PS plan. The collected data will serve as the input features for the ML model, which will be trained to accurately estimate the desired PS metrics (Targets) for a given state of the manufacturing cell. The trained ML model is then used within the MES system to facilitate online/dynamic scheduling of the production cell.

To provide a detailed description of the production scheduling (PS) environment considered in this research, Tables 1 and 2, together with Figure 2 are presented as an example of the fully-automated manufacturing cell at time point t. As depicted in Table 1, the example involves two jobs, J_1 and J_2, with the former having three operations (O_{11}, O_{12}, and O_{13}) and the latter having two operations (O_{21} and O_{22}). Furthermore, each operation can be performed on any of the three available machines (M_1, M_2 and M_3). Table 2 illustrates the operation sequence tuples that require a machine setup due to operational changes, such as a change in the mask (reticle). For instance, if operation four is scheduled after operation one on any machine, an additional setup is required to process it.

Figure 2 shows an arbitrary schedule for the production cell. Accordingly, as shown in Figure 2, at time point t each machine is blocked to process previously scheduled operations (for instance, M_1 will be

available at time point $t + b'_1$). Moreover, as defined in Table 2, to process operation five after operation two, a setup should be operated on M_1 that lasts for $S_{2,5}$ time units.

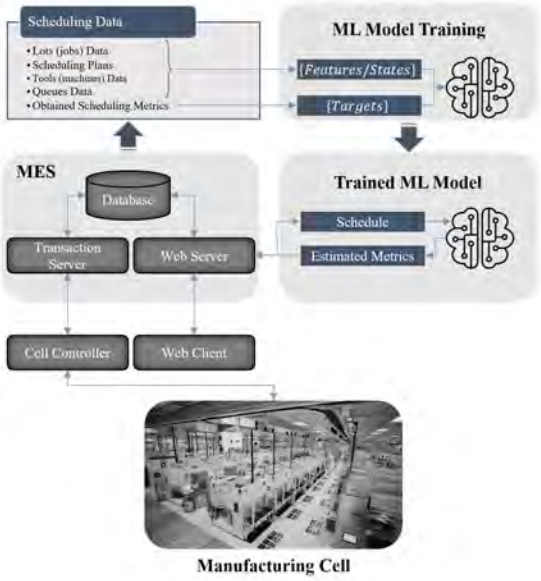

Figure 1: The General Framework Proposed in this Research.

Table 1: Flexible Job Shop production information.

job	operation (id)	machine set	job	operation	machine set
J_1	$o_{11}(1)$	M_2, M_3	J_2	$o_{21}(4)$	M_2, M_3
	$o_{12}(2)$	M_1, M_2		$o_{22}(5)$	M_1, M_2
	$o_{13}(3)$	M_1, M_3			

Table 2: Setup Required Operation Sequences.

setup required sequences
$(1,4),(1,3),(1,5),(2,4),(2,5),(3,5),(4,2)$

Overall, the considered FJS production environment in this research has specific features mimicking a real fully-automated production cell (such as photolithography) as follows:

- The production cell is observed at time point $t > 0$ through extracting data from the MES database (in real time).
- At time point t, each machine is not available for specific a period ($b' \geq 0$) due to processing running operations. This amount of time is stochastic since the variation is considerable but there is available historic data for it.
- There is a defined set of capable machines to process each operation, which is due to the required technology or quality.
- The processing time of each operation i on machine m (p'_{im}) and the required setup time to process operation i' after operation i ($s_{i,i'}$) are uncertain but there is available historical data to estimate them. Based on their considerable variability, they are both considered as stochastic values.
- The *Makespan* (C_{max}) refers to the completion time of the latest operation.

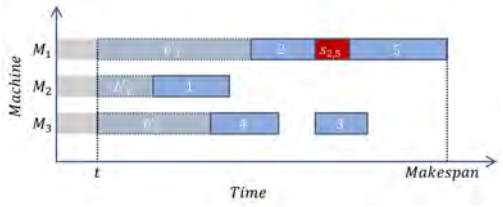

Figure 2: Proposed Flexible Job Shop Environment Example.

The following section details the simulation model developed to generate MES data dynamically for the described FJS production environment.

3 SIMULATION BASED MES DATA GENERATION

Table 3 details the parameters of the FJS environment considered in this investigation. These parameters are used to generate FJS instances and their simulation results according to the Makespan values. Consequently, the FJS test bed, in this investigation, consists of a manufacturing cell with eight machines (NM) and 12 jobs (NJ) waiting to be scheduled on these machines at time t. There are a total number of NO operations for these jobs that, for this illustrative example, follow the discrete uniform probability distribution $U(8\times NJ, 10\times NJ)$. Furthermore, the total number of sequences with setup times is equal to $0.5\times NO$. According to Ghasemi et al. (2020), the processing time (p') of operations in a photolithography workstation follows a gamma distribution (γ) with shape and scale parameters equal to 26.98 and 2.448, respectively. To reflect the features of a fully automated production cell, we consider the same distribution for processing times. To generate setup times (S) within the simulation model, we assume that each setup time follows a probability distribution with parameters $5\times\gamma(26.98, 2.448)$. Moreover, the initial scheduling block time (b') at time point t follows a mixture distribution of a discrete uniform distribution between 0 and 5 multiplied by the distribution of p'. Finally, to calculate the *Makespan* values for each FJS instance, 100 randomly independent simulation replications (SL) are executed.

Table 3: Considered FJS Environment.

NM	NJ	NO	NS	p'	S	b'	SL
8	12	$U(8\times NJ, 10\times NJ)$	$0.5\times NO$	$\gamma(26.98, 2.448)$	$5\times\gamma(26.98, 2.448)$	$U(0,5)\times\gamma(26.98, 2.448)$	100

In general, the simulation model acts as a function to estimate *Makespan* values for a given schedule as follows:

$$Makespan = F_{SL}(NM, NJ, NO, NS, p', S, b, schedule) \tag{1}$$

where, F_{SL} refers to the average of *Makespan* values obtained from SL simulation replications. As discussed above, executing simulation replications (experiments) in the case of complex manufacturing environments is usually highly time-intensive, which is not feasible for online decision making within the context of DTPS. Thus, in this research the main goal is to find a DL based metamodel with the following representation:

$$Makespan' = g(NM, NJ, NO, NS, p', S, b, schedule) \tag{2}$$

where, g refers to the trained ML model (DL model) that estimates *Makespan* values. Figure 3 shows an example of the generated training and testing data used to train and test the ML model. To distinguish different parts of the generated data, each part of the data is highlighted with different colors. As discussed in Section 2, the generated data consists of two parts ($\{Features/States\}$) and $\{Targets\}$. Here, Jobs Data, b' values, Setup Data, and Schedule are $\{Features/States\}$ and *Makespan* values are targets. Note that, in this example, due to space limitations we just provided *Makespan* values with $SL = 10$. Moreover the developed simulation model to calculate *Makespan* values is available at the following repository: GitHub

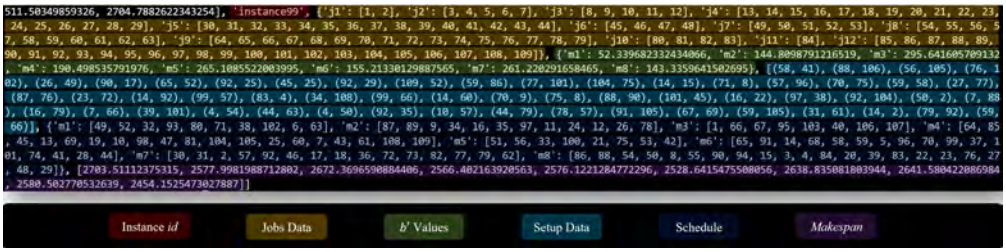

Figure 3: Generated MES Data Example.

Repository. The next section describes the developed ML model to find an accurate g (metamodel) in equation 2 to estimate *Makespan* values.

4 DEEP LEARNING-BASED META-MODELING

DL has revolutionized the field of ML by enabling the construction of models that can learn hierarchical and abstract representations of data (Goodfellow et al. 2016). The use of deeper layers allows for the capture of levels of abstract features from the input data, leading to better performance compared to traditional shallow ML models. Universal approximation theory (UAT) has provided a theoretical basis for understanding the capabilities of Artificial Neural Networks (ANNs) - the core of DL - to approximate any continuous function, provided that there are enough neurons (width and depth) and nonlinearity (Hanin and Sellke 2018; Lu et al. 2017). There are several architectures within DL that have been proposed to address specific challenges in modeling the structure of data. For example, convolutional neural networks are suitable for image data, while RNNs and Graph Neural Networks (GNNs) address processing sequences and graphs, respectively. Ultimately, powerful DL solutions are usually carefully designed architectures considering the underlying structure of their tasks. Accordingly, we propose to investigate the use of DL-based metamodels for the proposed FJS in this research. Specifically, to create a function g described in the previous section.

4.1 Conceptualization

4.1.1 Recurrent Neural Networks

RNNs are powerful for modeling sequential data through utilizing hidden states that are similar to memory while processing a sequence one by one in a recurrent manner resembling feed-back connection. The last hidden states can be used to represent the whole sequence. However, standard RNNs are prone to vanishing gradients and they suffer from inefficiencies for longer sequences (Hochreiter et al. 2001; Hochreiter et al. 2001). To overcome these limitations, Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) was proposed, which uses gating mechanisms to selectively retain or discard information in the hidden states during sequence processing. The gating mechanism allows LSTM to learn long-term dependencies and avoids the vanishing gradient problem. Another alternative to LSTM is Gated Recurrent Units (GRU) (Cho et al. 2014), which use a simpler architecture than LSTM, with only three gates instead of four. GRU usually have comparable performance to LSTM while they require fewer parameters and make them more efficient in some applications.

4.1.2 Attention Mechanism

The ability to adaptively attend to the most important elements of sets or sequences is achieved by means of the attention mechanism. It has been widely employed as a powerful component in DL. It can feature as a permutation invariant function in some cases, particularly when applied to a single sequence, which is called self-attention. Scaled dot-product attention creates normalized weights based on the query (Q) and key (K) tensors to be applied to the elements of the value (V) tensor to result in a representation for each element of query. Each tensor is a batch of elements that are vectors with the same dimension. Vectors

in Q and K have a dimension of d_k while V has a dimension of d_V. Accordingly, for each query vector, attention weights are created based on its dot product to the vectors of the key tensor that has the same number of elements as V. Multi-Head Attention (MHA) is its extension, where for each head different weights are learned, and their resulting representations are combined for the final representation. In fact, MHA divides the dimensions by the number of heads and creates an attention for each. Mathematically, MHA can be formulated as follows (Vaswani et al. 2017):

$$\begin{aligned} \text{Multi-Head Attention}(Q,K,V) &= (\Big\|_{i=1}^{h} head_i)\ W^O \\ head_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \\ \text{Attention}(q,k,v) &= \text{Softmax}(\frac{qk^T}{\sqrt{d_{k'}}})\ v \end{aligned} \tag{3}$$

Where $W_i^Q, W_i^K \in \mathbb{R}^{d_k x d_{k'}}$, and $W_i^V \in \mathbb{R}^{d_v x d_{v'}}$ ($d_{k'} = \frac{d_k}{h}$ and $d_{v'} = \frac{d_v}{h}$) are learned projection matrices, while $W^O \in \mathbb{R}^{d_v x d_v}$ is for combining the heads.

Both the scaled-dot attention and MHA have a linear form, but MHA can combine different attentions without increasing the number of parameters. The attention-based architectures have been effectively employed within different methods for sequences, particularly in natural language processing. Transformers (Vaswani et al. 2017) are a class of architectures that a combination of MHA modules, process sequences with a notable degree of success. Since attention considers every pair of elements inside the query and key tensors, it has a high computational complexity.

4.2 Model

Designing DL models is a challenging task, specifically when dealing with complex and hidden structures in the data such as complex PS environments. Moreover, combining different DL components can add another layer of complexity. Multi-layer perception (MLP), multiple consecutive fully-connected layers, learn a function between its input and output (Goodfellow et al. 2016). It can capture interactions among elements of its fix-length input but it is not permutation invariant. Furthermore, RNNs and attention mechanisms can be used for modeling sequences and interactions, respectively. Based on the characteristics of the FJS environment, we propose two architectures to estimate the *Makespan* as follows.

4.2.1 Rnn + Mlp

For the metamodel in PS, capturing the complex dependencies among elements of the FJS environment is essential. One way to achieve this is by creating a representation for each sequence, which can be further processed to model their interactions. To encode schedules for each machine, in this research, one RNN module is used. This RNN module after processing each operation within the sequence, memorizes its specific precedence. The initial state of the RNN is used to specify the machine and its ready-time (initial block times (b')). By concatenating the machine's id, which is a one-hot vector, with ready-time, a fully-connected layer creates the first hidden state for each machine. Moreover, the final hidden state contains information on the specific processing behavior of each machine along with its schedule. Another RNN is used to similarly encode the sequence of each job, without specific initialization. This is due to the fact that, for jobs, we only need to encode operations and their precedence. One of the key reasons to use shared RNN modules among machines and jobs is that it reduces the number of parameters and improves the model through parameter sharing.

In addition to creating representations for machines and jobs, a representation for the required setup tuples (i.e., i, i' in Table 2) is created. For this task a permutation-invariant function (Zaheer et al. 2017) that can handle variable number of elements is necessary. A fully connected layer is used to encode each

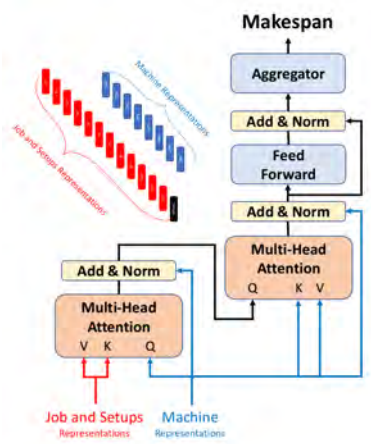

Figure 4: Architecture of the attention block used for the model of RNN + Attention.

pairwise sequence, and then the results are aggregated through a sum operation. In addition, the initial blockage of the machines (b') is concatenated with the respective machine representations, as it is an important feature. This is similar to the skip connection (He et al. 2016; Vaswani et al. 2017), which has been proven effective, especially for dealing with vanishing gradients in deeper architectures. After creating representations for eight machines, twelve jobs, and setup tuples, they are concatenated and then fed to an MLP. The MLP then utilizes every element to predict *makespan* after creating consecutive abstract representations.

4.2.2 Rnn + Attention

To reduce the high number of parameters and handle permutation invariance for jobs, we propose another architecture that incorporates the attention mechanism. The module can effectively attend to the most important elements and poses permutation invariance. We utilize MHA modules to combine identical representations that were created in the first architecture. In addition, the index of each machine is stacked with its respective representation. Similarly, a zero feature is concatenated to jobs and one to setups to ensure distinction between them for MHA. To model the dependencies, the architecture considers the attention of each machine on both jobs and setups. Based on them, queries are created for the second attention to determine how different machines can depend on each other while processing operations.

In the first MHA, machines are used as queries, while both jobs and setups are considered as keys and values. We create a residual connection for the resulting representations by adding machines, followed by a normalization layer (Ba et al. 2016; Ioffe and Szegedy 2015) to form queries for the second MHA. Then, we create another residual connection for the output of this attention by adding machines, followed by a normalization layer. We then apply two fully-connected layers (i.e., the feed-forward block), followed by another residual connection and normalization layer. we use a similar combination of residual connections and normalization layers to (Vaswani et al. 2017) that is proven to be effective. Finally, an aggregator creates the makespan based on the final representation that is created for each machine. One option is to use a fully-connected layer to project each representation to one dimension, followed by considering the maximum of them. Alternatively, an MLP can mix them and determine *makespan*. Although Max aggregation is a plausible option, the MLP-aggregator poses a more smooth behavior during training. The overall architecture of the attention block is presented in Figure 4 and shows how the two MHA modules based on Equation 3 are combined to make the final prediction.

5 EXPERIMENTS

To examine the efficiency of proposed DL-based metamodels in predicting *makespan*, this section presents the results obtained from the experiment. After generating 10,000 data instances (see Figure 3), we use it to form a benchmark. Consequently, the data were randomly split into three sets for training, validation, and testing of the models in an 8:1:1 ratio. We implemented the models using the *PyTorch* library and trained them with the *Adam* optimizer. To avoid overfitting, we used early stopping and dropout techniques (Goodfellow et al. 2016). The validation set is used for selecting the model to be evaluated in the test set. In fact, the selected model has the best performance in the validation set. We train the models for 200 *Epochs* and if the performance of the model in the validation set is not improved after 20 *Epochs* the training will be stopped (i.e., early stopping) and the model will be used for testing. The performance of the models are evaluated based on Root Mean Square Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). ML models are sensitive to hyperparameters and require careful optimization and extensive computational resources to select the best set of hyperparameters. Here, we report an 'out-of-the-box' configuration of the model (Yang et al. 2019), which is not necessarily the best possible performance of the model. The results obtained are presented in Table 4.

Table 4: Performance of 'out-of-the-box' models.

Model	MAPE	MAE	RMSE
GRU + MLP	0.0739	205.356	258.233
GRU + Attention	0.0692	194.834	251.706
LSTM + MLP	0.0723	201.893	256.346
LSTM + Attention	0.0696	193.794	251.91

The four models display an 'out-of-the-box' performance with an error rate of approximately 7% in predicting *makespan*. This result highlights the potential of DL as a viable solution. Moreover, the attention architecture (e.g., LSTM+Attention) exhibits a slightly better performance than the MLP model (e.g., LSTM+MLP). However, to make an accurate comparison of their superiority, it is necessary to conduct both hyperparameter optimization and cross-validation. The DL codes can be found through the following repository: GitHub Repository.

6 DISCUSSION

The main goal of this research is to examine the possibility of estimating PS metrics (*Makespan*, here) using ML-based metamodels (DL models, here). As discussed in Section 4.2, such a metamodel can help DTPS in providing online estimations of PS metrics while being integrated within the MES. Here, we discuss the key findings of this research:

- The DL-based metamodels developed here are only used to estimate *Makespan*, which is one of the most popular metrics, especially in the case of push manufacturing. Our experiments show that the metamodels developed based on DL are capable of estimating *Makespan* with less than 7.4% MAPE. Note that this result is obtained by 'out-of-the-box' configurations, i.e., without hyperparameter tuning that can significantly improve the performance of the model.
- Ghasemi et al. (2021) for the first time introduced the implementation of Learning-based Evolutionary Optimization Algorithms in PS problems. They considered a more simplified PS problem (typical stochastic job shop) in comparison with the current dynamic PS problem (FJS considered in this research). Their results showed 9% MAPE using Genetic Programming (GP) as the ML tool. This reduction in MAPE within a more complex PS environment shows the quality of the proposed ML-based metamodeling technique in the current research.
- The semiconductor frontend fab involves various complex machines and tools, and their availability can be unpredictable due to maintenance, breakdowns, or unexpected events. ML-based scheduling

methodology proposed in this research can potentially analyze historical data and provide predictions on equipment availability, enabling reactive/proactive scheduling adjustments to minimize downtime and optimize resource utilization.

7 CONCLUSIONS AND FUTURE WORK

In the context of Industry 4.0, this study introduces ML-based metamodeling technique for the dynamic scheduling of FJS production environments. Initially, a simulation model was employed to generate multiple instances of the FJS environment, emulating the behavior of the MES within a fully automated manufacturing cell. The resulting data was then used to train and test different DL models for predicting Makespan values. The statistical error metrics demonstrated the efficiency of the proposed DL-based metamodeling approach. However, it is important to note that this study examined only a specific FJS setting, as detailed in Section 2, and the DL-based metamodels were obtained without hyperparameter tuning. Worth nothing that in semiconductor environments, there is a constant flow of parts, while unexpected machine downtime or setup changes are common. These conditions introduce a high degree of uncertainty, which is not traditionally captured by a single objective function (considered in this research).

There are several promising directions for future research. One of the main objectives is to integrate the generated DL-based metamodel with an evolutionary optimization framework, as presented in Ghasemi et al. (2021), to create online scheduling agents for smart manufacturing cells. Another direction is to improve the Makespan prediction by adding into the metamodel the estimations calculated from analytical methods in a multi-fidelity framework (Lin et al. 2019). By doing so, we anticipate further enhancements in the performance of the scheduling system, ultimately leading to increased productivity and efficiency in manufacturing operations.

REFERENCES

Ba, J. L., J. R. Kiros, and G. E. Hinton. 2016. "Layer Normalization". *arXiv preprint arXiv:1607.06450*.

Cho, K., B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1724–1734. Association for Computational Linguistics.

Ghasemi, A., A. Ashoori, and C. Heavey. 2021. "Evolutionary Learning Based Simulation Optimization for Stochastic Job Shop Scheduling Problems". *Applied Soft Computing* 106:107309.

Ghasemi, A., R. Azzouz, G. Laipple, K. E. Kabak, and C. Heavey. 2020. "Optimizing Capacity Allocation in Semiconductor Manufacturing Photolithography Area – Case Study: Robert Bosch". *Journal of Manufacturing Systems* 54:123–137.

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

Hanin, B., and M. Sellke. 2018. "Approximating Continuous Functions by ReLU Nets of Minimal Width". *arXiv: 1710.11278*.

He, K., X. Zhang, S. Ren, and J. Sun. 2016. "Deep Residual Learning for Image Recognition". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Hochreiter, S., Y. Bengio, P. Frasconi, and J. Schmidhuber. 2001. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long Term Dependencies*, 237–243. Wiley-IEEE Press.

Hochreiter, S., and J. Schmidhuber. 1997. "Long Short-Term Memory". *Neural Computation* 9(8):1735–1780.

Ioffe, S., and C. Szegedy. 2015. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In *Proceedings of the 32nd International Conference on Machine Learning*, edited by F. Bach and D. Blei, Volume 37 of *Proceedings of Machine Learning Research*, 448–456. PMLR.

Jaskó, S., A. Skrop, T. Holczinger, T. Chován, and J. Abonyi. 2020. "Development of Manufacturing Execution Systems in Accordance with Industry 4.0 Requirements: A Review of Standard- and Ontology-Based Methodologies and Tools". *Computers in Industry* 123:103300.

Jeong, S.-h., G. Hwang, J.-Y. Lee, and J.-H. Han. 2023. "Machine Learning-Based Dispatching for a Wet Clean Station in Semiconductor Manufacturing". *Available at SSRN 4406114*.

Li, F., S. Lang, B. Hong, and T. Reggelin. 2023. "A Two-Stage RNN-Based Deep Reinforcement Learning Approach for Solving the Parallel Machine Scheduling Problem with Due Dates and Family Setups". *Journal of Intelligent Manufacturing*.

Li, Y., Z. Tao, L. Wang, B. Du, J. Guo, and S. Pang. 2023. "Digital Twin-Based Job Shop Anomaly Detection and Dynamic Scheduling". *Robotics and Computer-Integrated Manufacturing* 79:102443.

Lin, Z., A. Matta, and J. G. Shanthikumar. 2019. "Combining Simulation Experiments and Analytical Models with Area-Based Accuracy for Performance Evaluation of Manufacturing Systems". *IISE Transactions* 51(3):266–283.

Liu, M., S. Fang, H. Dong, and C. Xu. 2021. "Review of Digital Twin About Concepts, Technologies, and Industrial Applications". *Journal of Manufacturing Systems* 58:346–361.

Lu, Z., H. Pu, F. Wang, Z. Hu, and L. Wang. 2017. "The Expressive Power of Neural Networks: A View from the Width". In *Advances in Neural Information Processing Systems*, Volume 30, 1–9. Curran Associates, Inc.

Negri, E., S. Berardi, L. Fumagalli, and M. Macchi. 2020. "MES-Integrated Digital Twin Frameworks". *Journal of Manufacturing Systems* 56:58–71.

Negri, E., V. Pandhare, L. Cattaneo, J. Singh, M. Macchi, and J. Lee. 2021. "Field-Synchronized Digital Twin Framework for Production Scheduling with Uncertainty". *Journal of Intelligent Manufacturing* 32(4):1207–1228.

Stricker, N., A. Kuhnle, R. Sturm, and S. Friess. 2018. "Reinforcement Learning for Adaptive Order Dispatching in the Semiconductor Industry". *CIRP Annals* 67(1):511–514.

Takeda Berger, S. L., R. M. Zanella, and E. M. Frazzon. 2019. "Towards a Data-Driven Predictive-Reactive Production Scheduling Approach Based on Inventory Availability". *IFAC-PapersOnLine* 52(13):1343–1348.

Usuga Cadavid, J. P., S. Lamouri, B. Grabot, R. Pellerin, and A. Fortin. 2020. "Machine Learning Applied in Production Planning and Control: a State-of-the-Art in the Era of Industry 4.0". *Journal of Intelligent Manufacturing* 31(6):1531–1558.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. 2017. "Attention is All You Need". In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Volume 30. Curran Associates, Inc.

Villalonga, A., E. Negri, G. Biscardo, F. Castano, R. E. Haber, L. Fumagalli, and M. Macchi. 2021. "A Decision-Making Framework for Dynamic Scheduling of Cyber-Physical Production Systems Based on Digital Twins". *Annual Reviews in Control* 51:357–373.

Yang, K., K. Swanson, W. Jin, C. Coley, P. Eiden, H. Gao, A. Guzman-Perez, T. Hopper, B. Kelley, M. Mathea, A. Palmer, V. Settels, T. Jaakkola, K. Jensen, and R. Barzilay. 2019. "Analyzing Learned Molecular Representations for Property Prediction". *Journal of Chemical Information and Modeling* 59(8):3370–3388. PMID: 31361484.

Zaheer, M., S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. 2017. "Deep Sets". In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Volume 30. Curran Associates, Inc.

Zhang, F., J. Bai, D. Yang, and Q. Wang. 2022. "Digital Twin Data-Driven Proactive Job-Shop Scheduling Strategy Towards Asymmetric Manufacturing Execution Decision". *Scientific Reports* 12(1):1546.

AUTHOR BIOGRAPHIES

AMIR GHASEMI is an Assistant Professor in the department of IT & Logistics at Amsterdam School of International Business (AMSIB). His research interests include designing Simulation, Optimization, and Machine Learning-based Smart Agents in order to replace and/or support the human in decision making. His email address is: a.ghasemi2@hva.nl and his homepage is Link.

YAVAR TAHERI YEGANEH is a Research Fellow in the Department of Mechanical Engineering at Politecnico di Milano. He has a background in machine learning, computational physics, and robotics engineering. His current research interests include deep learning, reinforcement learning, graph neural networks, and digital twins. His email address is: yavar.taheri@polimi.it.

ANDREA MATTA is Professor of Manufacturing at Politecnico di Milano, where he currently teaches integrated manufacturing systems and manufacturing. His research area includes analysis, design, and management of manufacturing and healthcare systems. He is Editor in Chief of the Flexible Services and Manufacturing Journal. His email address is: andrea.matta@polimi.it.

KAMIL ERKAN KABAK is an Assistant Professor in the Department of Industrial Engineering, Izmir University of Economics. His research interests include simulation modeling and its application to complex manufacturing systems, stochastic processes and machine learning. His email address is: erkan.kabak@ieu.edu.tr.

CATHAL HEAVEY is an Associate Professor in the School of Engineering at the University of Limerick. He has published in the areas of queuing and simulation modeling. His research interests include simulation modeling of discrete-event systems; modeling and analysis of supply chains and manufacturing systems; process modeling; and decision support systems. His email address is cathal.heavey@ul.ie.