JYX
JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Kaikova, Olena; Terziyan, Vagan

Title: Deep Neural Networks, Cellular Automata and Petri Nets : Useful Hybrids for Smart Manufacturing

Year: 2024

Version: Published version

Copyright: © 2024 the Authors

Rights: CC BY-NC-ND 4.0

Rights url: https://creativecommons.org/licenses/by-nc-nd/4.0/

Please cite the original version:

Kaikova, O., & Terziyan, V. (2024). Deep Neural Networks, Cellular Automata and Petri Nets : Useful Hybrids for Smart Manufacturing. Procedia Computer Science, 232, 2334-2346. https://doi.org/10.1016/j.procs.2024.02.052

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 232 (2024) 2334–2346

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Industry 4.0 and Smart Manufacturing

Deep Neural Networks, Cellular Automata and Petri Nets: Useful Hybrids for Smart Manufacturing

Olena Kaikova [a], Vagan Terziyan [a,*]

[a] *Faculty of Information Technology, University of Jyväskylä, 40014, Jyväskylä, Finland*

Abstract

In the era of Industry 4.0 and beyond, intelligent and reliable models are vital for processes and assets. Models in smart manufacturing involve combining knowledge-based and data-driven methods with discrete and continuous modelling components. Formalism choice determines models' strengths and weaknesses in accuracy, efficiency, robustness, and explainability. Hybrid models seem to be the only way to address the complexity of modern industrial systems with respect to different and conflicting quality criteria. This study focuses on three paradigms: Petri nets, cellular automata, and neural network driven deep learning. We create four hybrids: Petri nets controlling deep neural networks, and vice versa; cellular automata controlling deep neural networks, and vice versa. These hybrids combine explainable discrete models with continuous black-box models, enhancing either explainability with robustness or elevating accuracy with efficiency. The flexibility of these and similar hybrids enable enhancement of the scope and quality of modeling and simulation in smart manufacturing.

© 2024 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 5th International Conference on Industry 4.0 and Smart Manufacturing

Keywords: Modelling; Petri nets; cellular automata; neural networks; hybrid models

1. Introduction

Reliable process models for design, control, operation and troubleshooting purposes are a key for efficient Industry 4.0 and smart manufacturing [1]. Taking into account the emerging "Industry 4.0 to Industry 5.0 transformation" challenge with human centricity and resilience in the focus [2], the traditional process modelling approaches, such as discrete-event modelling, must be enhanced with more sophisticated ones, including human behavior modelling

* Corresponding author. *E-mail address:* vagan.terziyan@jyu.fi

1877-0509 © 2024 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the scientific committee of the 5th International Conference on Industry 4.0 and Smart Manufacturing
10.1016/j.procs.2024.02.052

techniques. The latter require continuous computations and handling uncertainties when dealing with many social, cognitive, behavioral, ergonomic, and other human factors [3]. Therefore, the required models will be hybrid in their mathematical nature and will combine discrete and continuous modelling components with the autonomous (agent-based, behavioral, human) factors [4].

As mentioned in [5], a data-driven simulation model (e.g., digital twin design) requires machine learning and data mining for model extraction from data and its integration into the IoT system. According to [6], different modelling formalisms and frameworks have been developed to capture the discrete behavior of manufacturing systems. However, a purely discrete model does not capture the continuous variables of machine-level operation. Therefore, [6] suggests a framework for modeling and simulating manufacturing systems that captures the relationship between different quality indicators, which requires a hybrid model of machine dynamics combining discrete states and continuous variables. Continuous modelling frameworks are also either knowledge-based (e.g., physics-based modelling [7]) or data-driven, e.g., based on neural networks and deep learning [8], both of which have their strengths and weaknesses.

To overcome the limitations associated with purely knowledge-based and data-driven modeling frameworks, samples of hybrid models have been developed [9] and are still necessary, with improved model transparency, interpretability, and efficient analytics. Potential hybrids include: physics-informed machine learning [10], [11]; machine-learning-assisted simulation [12]; and explainable artificial intelligence [13]. These are expected to address smart manufacturing issues like product design, operation and maintenance, driven by comprehensive decision-making, while trading off between accuracy and explainability and dealing with both discrete and continuous simulations.

In [14], a hybrid has been defined as the result of merging two or more components of different categories to generate something new that combines the characteristics of these components into something more useful. Hybrid simulation (defined as a modelling approach that combines two or more of the following methods: discrete-event simulation, continuous system dynamics, and agent-driven system behavior) has recently experienced huge popularity [15]. According to [16], two (or potentially more) independent models A and B of different categories, being developed to address separate aspects of a problem with different formalisms, could be combined as a hybrid in one of these ways:

(a) *Sequential*: [Output (Model A) $\rightarrow$ Input (Model B)] $\Longrightarrow$ [Output (Hybrid) = Output (Model B)];
(b) *Interactive*: {[Output (Model A) $\rightarrow$ Input (Model B)] AND [Output (Model B) $\rightarrow$ Input (Model A)]} $\Longrightarrow$ {Output (Hybrid) = [Output (Model A) OR Output (Model B)}};
(c) *Mixed*: {Context I $\Longrightarrow$ [Output (Hybrid) = Output (Model A)]}; {Context II $\Longrightarrow$ [Output (Hybrid) = Output (Model B)]}; {Context III $\Longrightarrow$ [Output (Hybrid) = Output (Model A) $\oplus$ Output (Model B)]}.

We, however, will add here one important type of missing hybrid as follows:

(d) *Managed*: {[Output (Model A) $\rightarrow$ Configuration (Model B)] $\Longrightarrow$ [Output (Hybrid) = Output (Model B)]} OR {[Output (Model B) $\rightarrow$ Configuration (Model A)] $\Longrightarrow$ [Output (Hybrid) = Output (Model A)]}.

The latter (d) means that Model B is actually "does the job" while Model A controls the configuration (structure, hyperparameters, etc.) of Model B, or vice versa. Our former studies (summarized in [17]) considered such models as "meta-models" (i.e., "homogeneous hybrids" as defined in [18]) when models A and B are of the same category. Examples of these include semantic metanetworks [19], Bayesian metanetworks [20], and metapetrinets [21] among others.

In this paper, we are looking for heterogeneous hybrids of type (d). We consider A vs. B in the hybrids to be types of different modeling paradigms, either continuous vs. discrete modelling or vice versa. Our particular interest in this study focuses on hybrids, where Petri Nets or Cellular Automata represent the discrete modelling component within the hybrid (either managing one or being managed) and deep neural networks represent the continuous modelling component.

Petri Nets (PN) [22] are known to be a useful mathematical formalism (supported by various analysis and verification methods) for specification, discrete modelling and simulation of manufacturing systems. Consider, for example (among many), a PN model of a smart factory, which is proposed in [23] to support decision-making in mass customization. In the model, a token net is associated to the product and a system net to the facility. The resulting

model could be useful for analysis and performance evaluation, and, therefore, for decision support. A number of challenges regarding the PN formalism have been discussed in [24], which is a good review of real case studies within Industry 4.0. An important challenge related to the application of PNs for smart manufacturing is dealing with the deadlocks. Deadlocks are considered as undesirable states in a manufacturing system, which block its functioning. To deal with the deadlocks, either a system should be robust, i.e., constructed in such a way that the deadlocks cannot happen; or it should be resilient, i.e., capable of recovering from the deadlocks. These two options correspond to the two main approaches to the deadlock problem: their prevention or recovery. Automated manufacturing systems capable of deadlock prevention and recovery need special modified or hybrid PNs architectures, which are among the concerns of current research and development. Therefore, conclusions in [24] consider structural flexibility (reconfigurability) of PN models as an important future trend towards their further robustness, resilience, fault-tolerance, conflict-resolution, reachability, etc.

Cellular Automata (CA) [25] is another popular discrete modelling tool with a long history [26], which is used to model different and complex real life problems to facilitate their cost-effective physical implementation. CA are supposed to evolve over discrete space-time in single or multiple dimensions according to certain neighborhood-driven rules. CA are known to be massively parallel, homogeneous models with local interactions, and these models can be configured to behave as computationally universal simulators. The most famous CA of all times is Conway's "Game of Life" [27]. It is a two-dimensional CA, consisting of "living" and "non-living" cells. CA simulates artificial life by the rules: a living cell stays alive if there are exactly two or three living cells among the eight surrounding cells; fewer than two living neighbors will result in death through isolation; more than three living neighbors will cause death by overcrowding; and a non-living cell becomes alive if it has precisely three living neighbors. CA appear to be simple and efficient discrete simulators for a variety of tasks within industry, from the global to more focused simulations. For example, the conceptual model of industrial evolution presented in [28] is based on CA. In evolutionary simulation, the proposed model is able to appropriately explain the long-term evolution of industrial economic structures in both time and space, where CA representation has been used for land-use patterns and companies' spatial locations. The simulation results explain the competitive spatial organization of production in clusters of smaller companies. In another study presented in [29], CA has been used to accelerate the industrial information integration process by facilitating Big Data processing at low energy consumption in the Industry 4.0 scenario. Their study optimized the use of CA to explore true dynamics of modelled processes towards a cost-effective and "green" model for Industry 4.0. There were always attempts to add smartness and learnability to the CA framework, see e.g., learning CA [30].

Neural Networks (NN) and the modern Deep Learning framework [31] are the drivers of the technological revolution in data-driven continuous simulations for Industry 4.0 [32], [33]. Deep neural networks (and their variations) as the basis for simulation models' architecture show surprisingly high performance in modern industrial applications comparably to the former approaches [8]. However, this advantage comes at a high price, which is the lack of interpretability of such models and their behavior [34].

Objectives of our study presented in this paper include suggesting, designing, and studying potential applications for the following hybrid architectures: (1) PN controls configuration of NN; (2) NN controls configuration of PN; (3) CA controls configuration of NN; and (4) NN controls configuration of CA.

Because our focus is on the hybrids with the continuous vs. discrete modelling components, in this study we omit such hybrids as CA→PN and PN→CA, but interesting attempts to approach such hybrids are available in [35], [36], [37], and [38].

The following text of the paper is organized as follows: In Section 2, we present PN→NN and NN→PN hybrids; In Section 3, we present CA→NN and NN→CA hybrids; In Section 4, we discuss potential applications for hybrids; and we conclude in Section 5. Related work will be cited throughout the paper.

2. Hybrids of Petri Nets and Neural Networks

In this section, we are going to present two hybrid models where a discrete modelling component (PN) controls the configuration of a continuous modelling component (deep NN) and vice versa.

2.1. Petri net controls neural network

The goal of training NNs means finding such NN architecture (configuration, depth, hyperparameters, etc.) that being trained will generalize well on the given dataset (i.e., will perform well on training data and then passes tests on previously unseen data). However, while trying too hard to learn different features from the dataset, NNs often overfit, i.e., in addition to learning the main concept, they become biased towards possible statistical noise in the dataset. Various regularization techniques have been used to address overfitting, among which one of the most common and efficient is dropout [39]. Dropout means randomly dropping out (according to defined dropout probability) the neurons (usually from hidden layers) in NN so that all the forward and backwards connections of these neurons are also removed, thus creating a new (more modest) network configuration from the parent network. Dropout is performed during NN training so that NN configuration is different during each training batch. This ensures that the model is getting generalized and hence reducing possible overfitting.

The randomness of the dropout process [40] makes the job but leaves some uncertainty in the final generalization success because it keeps the data scientist out of the NN training process control loop. We would like to suggest a way to explicitly control the dropout process by using PN (this subsection) or CA (subsection 4.1). The proposed hybrid model for this purpose is shown in Fig. 1.

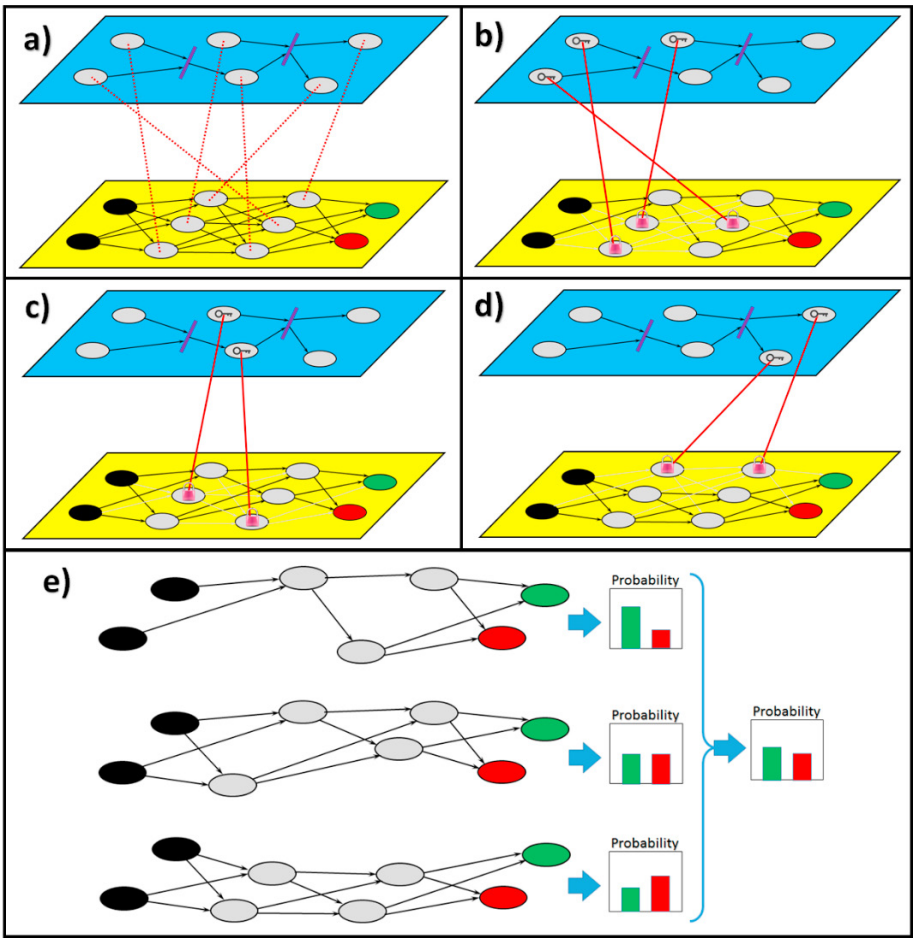

Fig. 1. A hybrid, which implements a PN-driven drop-out control for a deep NN is illustrated: (a) places of the PN from the upper layer correspond to the hidden neurons of the NN from the lower layer; (b) the initial PN marking with the tokens ("keys") unlocks the corresponding neurons from the lower layer NN (i.e., these with corresponding keys will be "unlocked" and become active for training while others will be removed from current NN configuration); (c) PN generates its new marking due to fired transitions and, therefore, new neurons will be unlocked by the tokens (keys) while others locked (removed); (d) the last cycle of PN marking and corresponding NN configuration; (e) all three configurations are making decision independently and their result is integrated into final probability distribution among the classes behind the output neurons.

The hybrid model (Fig. 1) has two layers: the upper one or the controlling layer is a PN; and the lower one or the controlled layer is a NN. The places of the controlling PN correspond to the hidden neurons of the controlled NN (Fig. 1a). A neuron from the NN configuration is considered as active (i.e., it is "unlocked" and is taking part in the current NN configuration) if there is a token ("key") in the corresponding place of the PN; otherwise the neuron is "locked" or dropped-out from the current configuration (Fig. 1b). PN behaves by changing its marking due to firing its transitions (Fig. 1c, d). Accordingly, the NN is changing its configuration because some neurons will be unlocked and some locked back (dropped out). Resulting configurations are trained independently with the particular batches of data and take part in the final decision as an ensemble (Fig. 1e). This process means that, instead of a random choice of the dropped-out neurons for each configuration, we have the controllable (by PN) dropout process.

2.2. Neural network controls Petri net

Here we suggest a hybrid model architecture (see Fig. 2) where some PN simulates a complex logistic process and NNs control the PN structure. These NNs are trained to automatically detect (recognize, classify) various items appearing in the process, make decisions on corresponding structural changes needed within the PN architecture to handle the recognized items and enforce the decided changes in run-time. Fig. 2a illustrates the way NNs interact with PN in the hybrid model. Certain transitions in PN can be "lockable", i.e., when locked (dropped-out) they do not fire. The status of such transitions ("locked" or "unlocked") is defined by the so-called metatransition. Metatransition has one input place (metaplace). Tokens, which may appear in the metaplaces, are attributed to particular (physical) items, which are defined by a set of measurable parameters (continuous attributes of tabular or image data formats). Metatransition works as a NN, i.e., takes the vector of an item's parameters as an input and outputs the decision as a status ("locked", "unlocked") for a corresponding PN transition. In this way, depending on what kind of item is "travelling" through the PN structure, the particular (unlocked) PNs substructure will be taking care of its further logistics. Such a hybrid automates (due to NNs) tagging of items (aka "coloring" them according to the "colored PN" terminology) and adapting (personalizing) the PN structure to the particular items. One particular use-case scenario (Fig. 2b), where NN controls logistic structure based on the recognition of items observed through the camera, will be discussed in Section 4.

3. Hybrids of Cellular Automata and Neural Networks

In this section, we are going to present two hybrid models where a discrete modelling component (CA) controls the configuration of a continuous modelling component (deep NN) and vice versa.

3.1. Cellular automata control neural network

The hybrid presented here has similar objective to the one from subsection 2.1, i.e., discrete and explainable control of the dropout regularization for NNs. However, here we have different instruments for approaching the objective, i.e., CA instead of PN. The unique specifics of CA makes the control itself very different from the one applied to PN. Fig. 3 illustrates the hybrid. The deep NN in the figure performs the main modelling task while the CA is controlling its structure to achieve better generalization performance of the NN. The cells in CA correspond to the neurons in NN and interact in a way (see Fig. 3a) that: if the cell (aka the "locking key") is initialized or changed to "alive" status, then the corresponding neuron of the NN will be dropped-out ("locked" or "frozen"); if the cell "dies" (the key disappears), then the corresponding neuron of the NN will be activated (dropped-back or "awakened"). In this way, every new generation of living cells (which appears following the CA rules), defines the currently operating substructure of NN. Fig. 3b demonstrates the part of the CA process (four generations) operating according to Conway's "Game of Life" rules [27]. One may see that each generation locks (unlocks) corresponding neurons of the NN. Therefore, it implements the dropout process where four different NN sub-architectures (as a kind of ensemble) will potentially generalize better than the complete NN can do. CA, in this case, works like an explainable controller, i.e., one can consciously change its impact by defining the initial generation of living cells or by changing the rules.

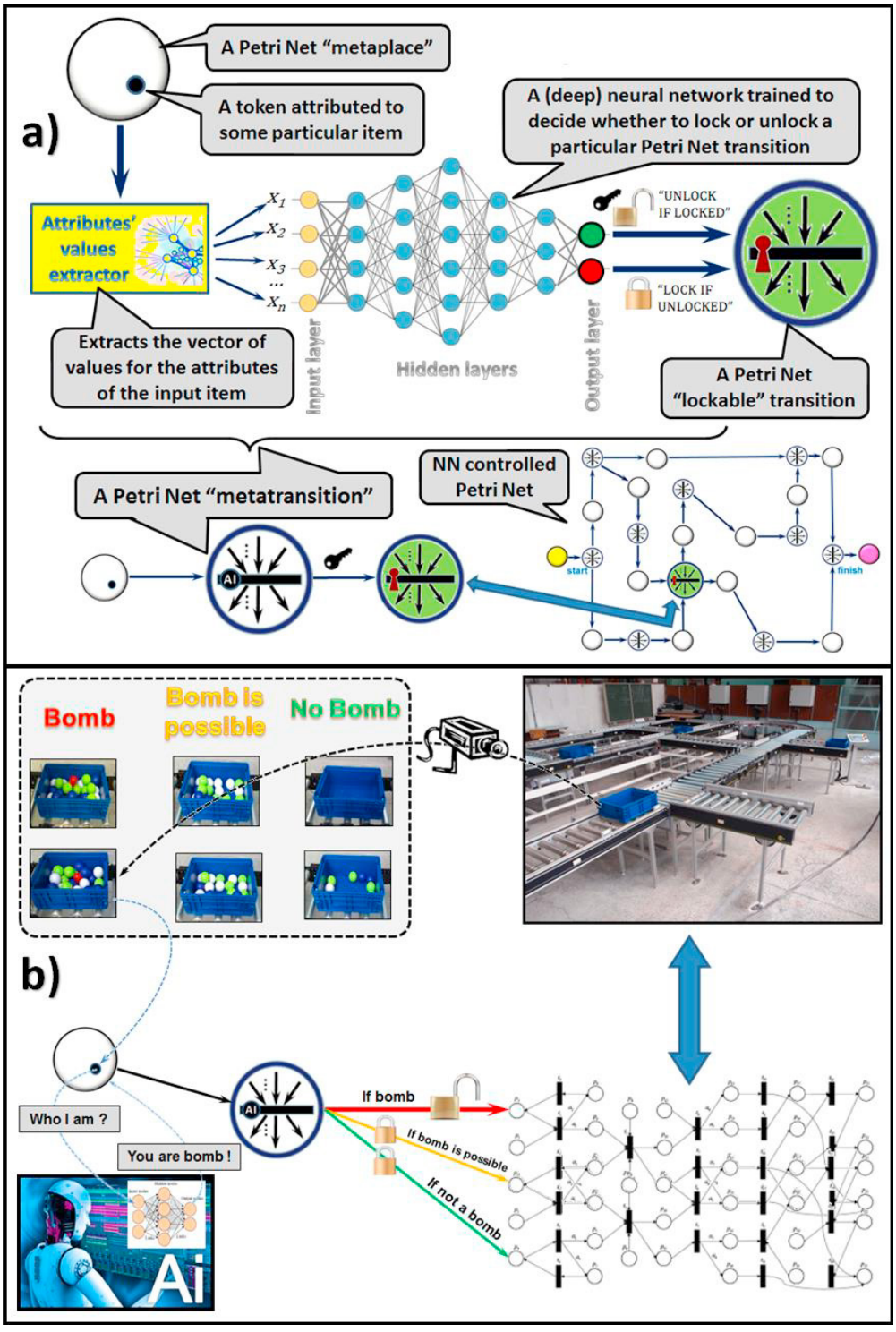

Fig. 2. Neural-Network-driven Petri Net process control illustrated: (a) some PN tokens can be attributed to particular physical items described by a vector of measurable attributes. Some PN transitions can be locked (dropped-out) or unlocked (dropped-back). Decisions on the transition status for lockable transitions are made by NN after processing such tokens' attributes; (b) example scenario where the camera observes each item and NN decides on further logistics to be applied to handle the item (i.e., some parts of PN architecture, which simulates the logistics, will be locked).

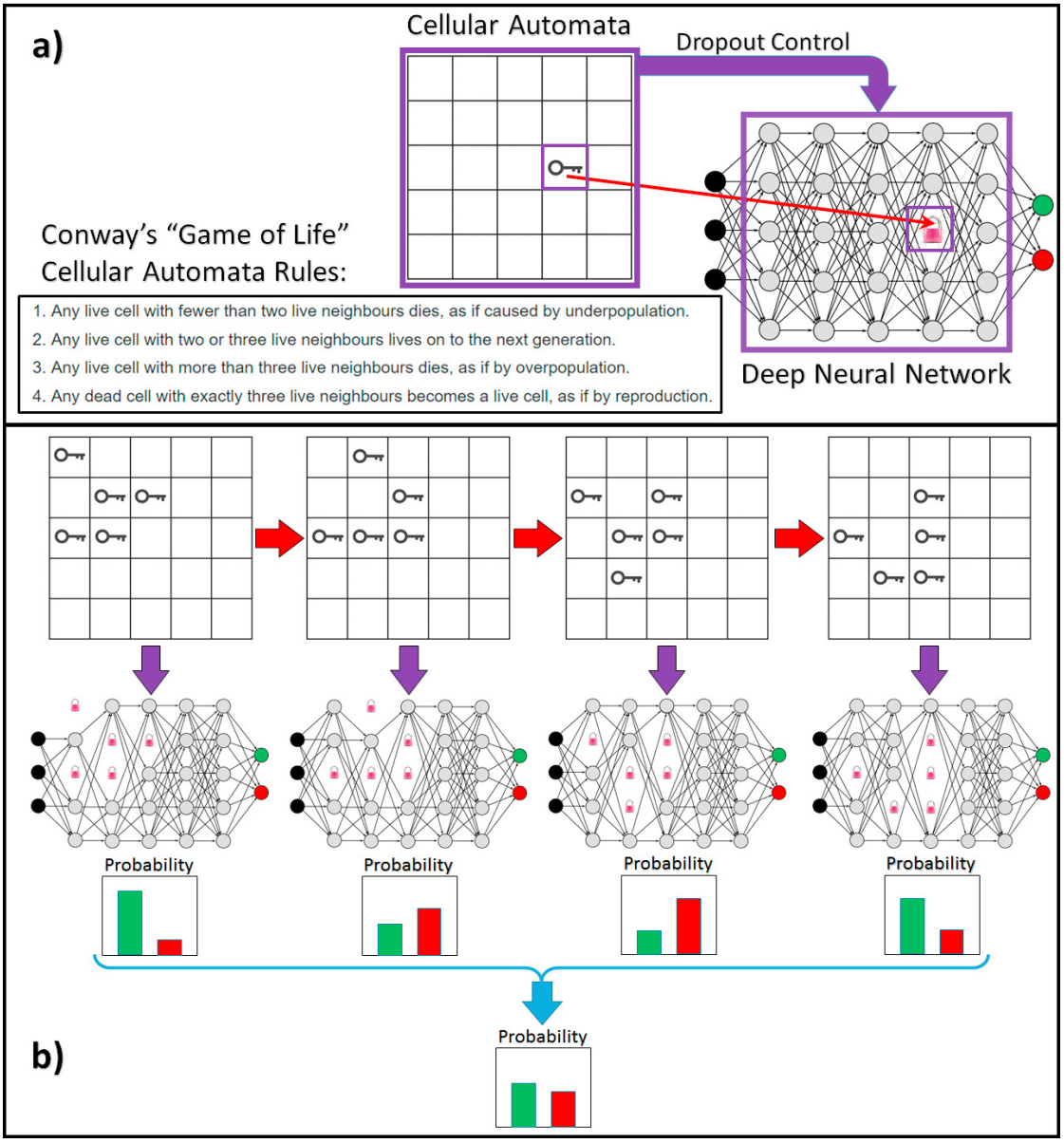

Fig. 3. Cellular-Automata-driven dropout control in deep NN is illustrated: (a) cells in CA are attributed to the neurons in NN so that the status "alive" for each cell (marked with the "key") means status "locked" (dropped-out) for the corresponding neuron; (b) example of CA evolution (with respect to the rules) and corresponding changes enforced in the NN structure. This in fact controls the dropout process, making the regularization (for better NN generalization performance) benefit from conscious control instead of randomness.

3.2. Neural network controls cellular automata

Here the basic modelling layer of the hybrid architecture is CA, which simulates a dynamic evolutionary process by applying the defined set of rules. The NN-driven layer is used on top of CA, aiming to change the rules (to switch from one set of rules to another one in the run-time) depending on the observed and changing context. Therefore, CA can simulate quite complex dynamics, where the modelling layer is locally simple and explainable and the controlling layer is learnable and adaptive to more global context changes. The hybrid architecture is illustrated in Fig. 4.

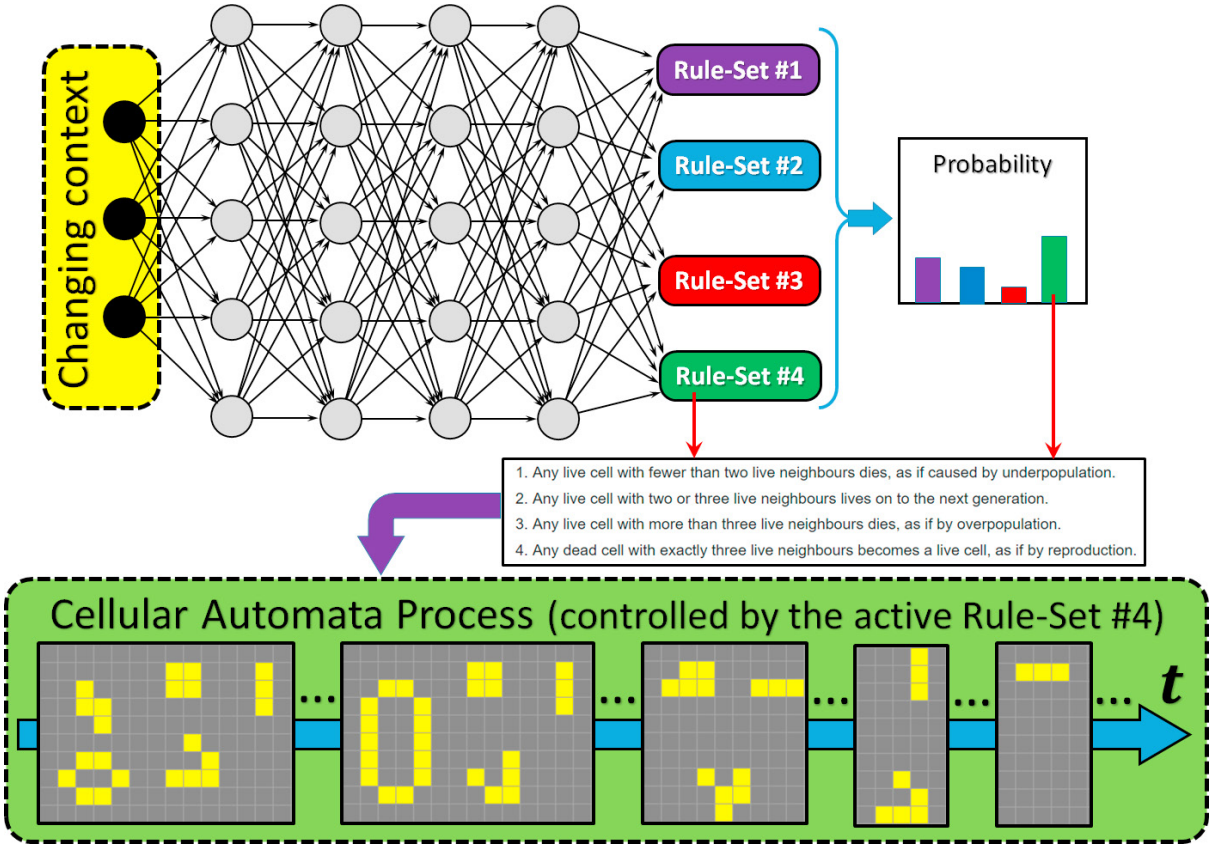

Fig. 4. Neural-Network-driven Cellular-Automata process control illustrated. One may see that CA-driven process simulation is generated and evolving according to the set of rules chosen by the NN on top of it. In the figure, the currently active set of rules (#4) is the one from Conway's "Game of Life". However, if context changes and the NN gives advantage to another set of rules, then the process dynamics will change in the run-time accordingly.

4. Discussion

We have presented several possible options for the hybrid modelling architecture, which combines continuous and discrete modelling components. We construct the hybrids from two components in a way that one component controls the configuration or behavior (evolution dynamics) of another one. In this section, we are going to discuss potential application areas and use cases for these hybrids in the context of Industry 4.0 and smart manufacturing.

Taking into account that chosen components (PNs and CA for discrete modelling) and NNs for continuous modelling have their special features (also pros and cons), the hybrids constructed from these components may have interesting and useful properties.

PNs are good for simulating, visualizing, analyzing consistency and optimizing (with expert support) the dynamic (logistics) processes. One may easily see how the items are "travelling" through the PN model structure following the transitions' logic. Experts can relatively easily observe the simulations, understand what is ongoing there and make needed changes consciously. However, PNs are not capable of learning or making intelligent decisions. They are not capable of self-management (particularly self-configuration) for, e.g., automatically recovering from possible deadlocks. This may limit their use in Industry 4.0 and smart manufacturing. PNs are good as decision-support tools but not as decision-making ones.

CA is a good tool for modelling evolutionary non-linear (spatially and temporally discrete) dynamics following state update functions or transition rules, and they can emulate anything computable like a universal Turing machine (i.e., an appropriate selection of initial conditions can ensure that the system carries out arbitrary algorithmic

procedures). It is known that quite simple CA rules can result in quite complex dynamics. However, approaching the rules from the observed dynamics of the real industrial systems is not an easy task. Typical industrial use cases include physical processes simulation and analysis (such as gas or fluid dynamics, morphological growth of various materials), energy consumption, business evolution simulation, secure data flow through advanced cryptography, etc. However, CAs themselves cannot yet learn and make decisions on reconfiguration (if and when necessary) of the rule set which they follow. CA can simulate complex systems (even intelligent ones) and it can be used as a decision-support tool (with informative and explainable visualization). However, it cannot be considered as an intelligent modelling framework or automatic decision-making tool yet. CA would definitely benefit from the development of self-optimization techniques on automatic designing and self-configuration of the local rules in accordance with the particular real world industrial phenomena.

NNs are good when the objective is a decision-making model and the model is automatically constructed (by learning) based on training data (with mainly continuous attributes). Mathematical fitness to data without real understanding of the nature behind it makes NNs, on the one hand, highly performed (good accuracy) in making decisions, however, on the other hand, incapable of giving any clue about how and why just these decisions have been chosen (lack of interpretability and explainability). Therefore, NNs is a good (also intelligent and learnable) tool for automatic decision-making (when appropriate) but not for such responsible decision-support tasks where human experts need to see an interpretable simulation model and understand its outcomes.

Therefore, the expectations regarding the hybrids presented in this paper is that they will inherit the strengths of the component architectures, i.e.: intellect and high decision accuracy from NNs; together with simulation effectiveness and explainability from PNs and CA).

The hybrids PN→NN and CA→NN (with the generic slogan "towards improved manageability and explainability of the black-box decision models") are designed to support general decision-making objectives and the NN component guarantees that. However, PN or CA (each with its own flavor) add some interpretable (explainable) control option (particularly dropout control) to manage the way NN is being trained. This means that the dynamics and evolution of some real industrial processes (simulated by PN or CA) can directly influence the processes of finding an optimal NN architecture (dropout as a NN's structure learning and regularization process) to make better decisions within particular decision points of these processes. As a summary, we may say that such a couple of hybrids provide some explainable control option for human experts (i.e., industrial domain experts rather than data scientists) to manage high-performing but "black-box" decision models in smart manufacturing. We believe that hybrid architectures, PN→NN and CA→NN, where Petri nets and cellular automata control dropout in neural networks, offer intriguing possibilities for improving the performance of dropout techniques. These combinations of different computational paradigms can potentially find applications in various Industry 4.0 contexts where advanced data analysis and decision-making are crucial. Here there is a couple of industrial applications where such hybrids may work effectively: (a) manufacturing process optimization – by integrating real-time data from sensors with the NN's decision-making process, the system could dynamically adjust dropout policies based on the current state of the process (defined by PN or CA), leading to higher NN's accuracy and, therefore, to enhanced process efficiency and reduced defects; (b) similarly in predictive maintenance NNs could adapt dropout based on historical maintenance records and sensor data, resulting in more precise predictions, etc.

The hybrids NN→PN and NN→CA (with the generic slogan "bringing more intelligence and learnability into the discrete simulation models") are designed for modelling, simulation and decision-support objectives and either the PN component or the CA component guarantee that. However, the NN component on top embeds some learnable intelligence into the basic models. These hybrids enable "smarter" self-configurable PNs or CA based on the decisions automatically made by NN. This means that additional intelligence (learnable from data as a NN) may enhance the decision points within industrial processes previously managed by humans or hardcoded by the fixed rules. In this way, the decision-making power of PNs and CA will grow due to the known high decision performance of NNs. Therefore, we may say that the hybrids NN→PN and NN→CA enable discrete simulation models (PN or CA) to benefit from the intelligence and learnability of neural networks. The combination of the two components allows for more adaptive, data-driven decision-making, leading to increased efficiency, reduced costs, and improved performance in various industrial processes and decision-making. Here are some concrete industrial application cases where such hybrids could be a reasonable option: (a) manufacturing – NNs learn from historical data and make intelligent decisions

while the discrete simulation models (PN or CA) can dynamically self-configure and adapt to changing production conditions. Such a flexibility enables efficient processes, improved production rate, and reduced operational costs; (b) supply chain and logistics – NN component learns from historical supply chain data, allowing the discrete simulation models to dynamically adjust parameters and make better-informed decisions related to inventory management, demand forecasting, and order fulfillment. This adaptability leads to more efficient and responsive supply chain operations; (c) smart grid control – integrating NNs with the discrete simulation models, power distribution networks can adaptively adjust their configuration and resource allocation based on real-time data. This enables more efficient load balancing, fault detection, and demand prediction, leading to better energy management and reduced energy wastage; (d) predictive maintenance of smart assets – NN component can learn from sensor data and historical maintenance records to predict equipment failures. The discrete simulation models can then automatically adjust maintenance schedules, optimize spare parts inventory, and allocate maintenance resources efficiently, reducing corresponding costs.

We experimented with all the four hybrids within the project IMMUNE: "Cyber-Defence for Intelligent Systems", which is a NATO SPS project (http://recode.bg/natog5511). Various scenarios have been simulated within the special laboratory (aka cyber-physical environment), which enables combining digital simulations with a real physical environment, including complex item storage and logistics. An interroll cassette conveyor (Fig. 2b) is part of the laboratory infrastructure and it is used to simulate various scenes (cassette loads and their further delivery logistics) simulated by either PNs or special CA and monitored by cameras. Independently designed PN and CA simulations were enhanced with the (convolutional) NN component capable of classifying observed cassette loads and automatically reconstructing the delivery plan (PN) or transformation rules (CA) accordingly in a run time.

It is important to mention that we designed complex experiments to test couples of hybrids simultaneously. For example, one of such was the $PN_b \rightarrow NN \rightarrow PN_a$ experiment chain, one of which parts acts as a $NN \rightarrow PN_a$ hybrid and implements a self-configurable (driven by NN) logistics simulated by PN_a; and the NN itself is also part of another hybrid $PN_b \rightarrow NN$ where another PN_b controls the dropout regularization of the NN training process. In this way, two hybrids have been tested with the same experiment. Other experimental chains were as follows: $CA_b \rightarrow NN \rightarrow CA_a$; $PN \rightarrow NN \rightarrow CA$; $CA \rightarrow NN \rightarrow PN$.

During the experiments, certain initializations and configurations for PNs (or rules for CA) has been found for the $PN \rightarrow NN$ and $CA \rightarrow NN$ hybrids, such that NN with controlled dropout performs significantly better than with randomness-driven dropout. It also has been observed that $NN \rightarrow PN$ and $NN \rightarrow CA$ hybrids (when NN is trained) correctly produce PN or CA sub-configurations (or rules) for handling specific cases in a more flexible way if you compare it with the complete and fixed configurations.

These were just preliminary experiments with relatively small numbers of training data (pre-trained models were enhanced with an additional 2200 images). We believe, however, that the full hidden potential of the hybrid architectures could be much greater within a real industrial environment and with bigger volumes of training data.

Such or similar architectures could be useful in the cases where the discrete simulation logic and complex logistics is supposed to be combined with automated decision-making aiming reasonable trade-off between decision explainability and accuracy. Potential areas include but not limited to: rooting, resource allocation, capacity optimization, etc., in telecommunication (optical, wireless, hybrid) networks (see Fig. 5 and https://cordis.europa.eu/project/id/101008280); human activity recognition in industrial work processes, as it is shown in [41], or for designing digital twins for fault diagnostics and prediction, see, e.g., [42].

In addition, in our study, we employed few datasets related to manufacturing process optimization and predictive maintenance of smart industrial assets (paper industry and power transmission via smart grids) collected by our former industrial projects (http://www.cs.jyu.fi/ai/SmartResource_UBIWARE.html) SmartResource and UBIWARE. We used these particularly to check the performance of our hybrid architectures, where CA and PN are employed as regularization approaches for the NN in comparison to random dropout. For example, in binary classification case (healthy vs failure state) with about 5000 samples and 15 features, we achieved 0.88 ($CA \rightarrow NN$) and 0.87 ($PN \rightarrow NN$) accuracy rates in comparison to 0.85 (random dropout); 0.89 ($CA \rightarrow NN$) and 0.88 ($PN \rightarrow NN$) precision vs 0.87 (random dropout); 0.87 ($CA \rightarrow NN$) and 0.86 ($PN \rightarrow NN$) recall vs 0.83 (random dropout); and 0.88 ($CA \rightarrow NN$) and 0.87 ($PN \rightarrow NN$) F1 score vs 0.85 (random dropout). These results demonstrate that utilizing CA and PN as regularization mechanisms in neural networks leads to improved performance in the context of Industry 4.0

applications. The incorporation of CA and PN enables NN to control dropout adaptively, enhancing the model's generalization and robustness to varying data distributions and complexities. However, the effectiveness of our hybrid architectures may vary with different network architectures and dataset characteristics, and further investigations are foreseen, which may provide additional insights.

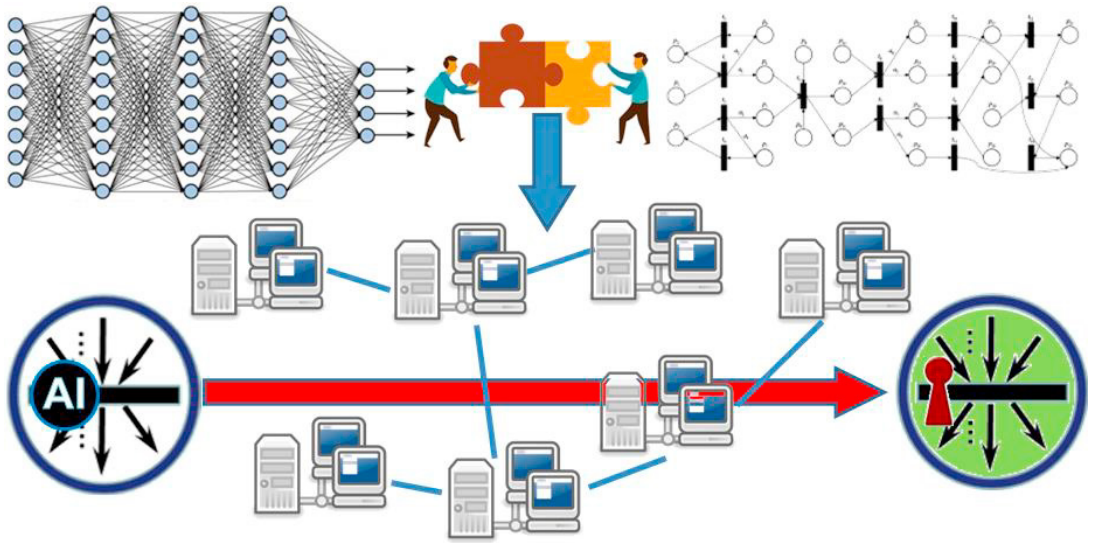

Fig. 5. Generic schema of potential application of the NN →PN hybrids to management of telecommunication networks (e.g., rooting, resource allocation, optimization, etc.). Activity in such networks could be simulated with the PN-driven modelling component, while a suitable configuration depending on changing context could be automatically decided and controlled by the NN-driven component.

5. Conclusions

Hybrid approaches, which aim to (a) either intellectualize traditional discrete simulation models [43]; or (b) make intelligent (deep learning driven) models explainable [44], are becoming a trend due to the emergent Industry 4.0 needs. Hybrid architectures suggested in this paper are layered, assuming that the modelling component from the lower layer simulates a particular real-life process and the modelling component from the upper layer controls configuration and evolution dynamics of the lower layer. We limit the study by the cases when the components collaborating in this way belong to different modelling categories: either a discrete modelling component (PN or CA in this paper) controls NN as a learnable and continuous modelling component or vice versa. Therefore, we have considered four types of such hybrids, particularly: (a) PN→NN; (b) NN→PN; (c) CA→NN; and (d) NN→CA. Hybrids (a) and (c) apply discrete control over the configuration of NN to add some explainability to the NN (i.e., consciously finding optimal configuration of the NN for better generalization performance). Hybrids (b) and (d) aim to intellectualize discrete simulation models, such as PN or CA. These hybrids apply NN decision-making to reconfigure automatically and in run-time either PN (change operational structure) or CA (change operational rules) depending on the context. Preliminary experiments (with the four hybrids) show that explainable control over NN or embedded (into PN or CA) NN intelligence provide additional performance and simulation flexibility to the modelling components of the hybrids.

Actually, the considered four hybrid architectures are the classes of many potential hybrids. This is because these modelling components (PNs, CA, and NNs) may have very different categories themselves, e.g.: PNs can be deterministic, stochastic, colored, high-level, fuzzy, temporal, etc. [45]; CA can be deterministic, probabilistic, non-uniform, reversible, higher-order, totalistic, partitioned, etc. [46]; NNs can be feedforward, recurrent, residual, convolutional, autoencoder, adversarial, etc. [47]. Each combination of these options gives a valid and specific hybrid, which can be used for specific industrial simulations.

We believe that each of the four hybrid architectures (PN→NN, CA→NN, NN→PN, and NN→CA) represents a significant leap beyond the state-of-the-art in the current modelling frameworks. By combining diverse computational

paradigms and leveraging the strengths of both discrete simulation models and neural networks, these innovations pave the way for more intelligent, adaptable, and data-informed decision-support systems in various industrial domains. The ability to integrate learnable intelligence into discrete models enhances their capabilities, leading to improved efficiency, accuracy, and performance, and fostering the advancement of Industry 4.0 and beyond.

The future work for these hybrid architectures holds exciting possibilities to further advancing the field of intelligent decision-support systems in various industrial domains. By leveraging the strengths of both discrete simulation models and NNs, researchers can unlock novel opportunities for more efficient, adaptable, and data-informed decision-making, contributing to the advancement of Industry 4.0 and beyond. Particularly, in our future studies, we are going to develop a metric capable of evaluating the quality of different hybrids based on conflicting qualitative and quantitative criteria. In addition, we plan to transfer our experimental hybrid models built within the industrial laboratories to the current industrial processes.

References

[1] Chinda, R., Ponsatorn, R., Anantpinijwatna, A., Pessoa, F. P., Woodley, J. M., and Mansouri, S. S. (2019). "Process model validation and analysis for intensification of an industrial scale process". In: *Computer Aided Chemical Engineering* (Vol. 46, pp. 955-960). Elsevier. https://doi.org/10.1016/B978-0-12-818634-3.50160-0

[2] Golovianko, M., Terziyan, V., Branytskyi, V., and Malyk, D., (2023). "Industry 4.0 vs. Industry 5.0: Co-existence, transition, or a hybrid". *Procedia Computer Science*, **217**: 102-113. Elsevier. https://doi.org/10.1016/j.procs.2022.12.206.

[3] Longo, F., Nicoletti, L., and Padovano, A. (2019). "Modeling workers' behavior: A human factors taxonomy and a fuzzy analysis in the case of industrial accidents". *International Journal of Industrial Ergonomics*, **69**: 29-47. https://doi.org/10.1016/j.ergon.2018.09.002

[4] Longo, F., Nicoletti, L., and Padovano, A. (2019). "Emergency preparedness in industrial plants: A forward-looking solution based on industry 4.0 enabling technologies". *Computers in Industry*, **105**: 99-122. https://doi.org/10.1016/j.compind.2018.12.003

[5] Friederich, J., Francis, D. P., Lazarova-Molnar, S., and Mohamed, N. (2022). "A framework for data-driven digital twins for smart manufacturing". *Computers in Industry*, **136**: 103586. https://doi.org/10.1016/j.compind.2021.103586

[6] Saez, M., Barton, K., Maturana, F., and Tilbury, D. M. (2022). "Modeling framework to support decision making and control of manufacturing systems considering the relationship between productivity, reliability, quality, and energy consumption". *Journal of Manufacturing Systems*, **62**: 925-938. https://doi.org/10.1016/j.jmsy.2021.03.011

[7] Willcox, K. E., Ghattas, O., and Heimbach, P. (2021). "The imperative of physics-based modeling and inverse theory in computational science". *Nature Computational Science*, **1(3)**: 166-168. https://doi.org/10.1038/s43588-021-00040-z

[8] Kusiak, A. (2020). "Convolutional and generative adversarial neural networks in manufacturing". *International Journal of Production Research*, **58(5)**: 1594-1604. https://doi.org/10.1080/00207543.2019.1662133

[9] Wang, J., Li, Y., Gao, R. X., and Zhang, F. (2022). "Hybrid physics-based and data-driven models for smart manufacturing: Modelling, simulation, and explainability". *Journal of Manufacturing Systems*, **63**: 381-391. https://doi.org/10.1016/j.jmsy.2022.04.004

[10] Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). "Physics-informed machine learning". *Nature Reviews Physics*, **3(6)**: 422-440. https://doi.org/10.1038/s42254-021-00314-5

[11] Kim, S. W., Kim, I., Lee, J., and Lee, S. (2021). "Knowledge integration into deep learning in dynamical systems: An overview and taxonomy". *Journal of Mechanical Science and Technology*, **35(4)**: 1331-1342. https://doi.org/10.1007/s12206-021-0342-5

[12] Azab, E., Nafea, M., Shihata, L. A., and Mashaly, M. (2021). "A machine-learning-assisted simulation approach for incorporating predictive maintenance in dynamic flow-shop scheduling". *Applied Sciences*, **11(24)**: 11725. https://doi.org/10.3390/app112411725

[13] Ahmed, I., Jeon, G., and Piccialli, F. (2022). "From artificial intelligence to explainable artificial intelligence in industry 4.0: a survey on what, how, and where". *IEEE Transactions on Industrial Informatics*, **18(8)**: 5031-5042. https://doi.org/10.1109/TII.2022.3146552

[14] Mustafee, N., Brailsford, S., Djanatliev, A., Eldabi, T., Kunc, M., and Tolk, A. (2017). "Purpose and benefits of hybrid simulation: contributing to the convergence of its definition". In: *Proceedings of the Winter Simulation Conference* (pp. 1631-1645). IEEE. https://doi.org/10.1109/WSC.2017.8247903

[15] Brailsford, S. C., Eldabi, T., Kunc, M., Mustafee, N., and Osorio, A. F. (2019). "Hybrid simulation modelling in operational research: A state-of-the-art review". *European Journal of Operational Research*, **278(3)**: 721-737. https://doi.org/10.1016/j.ejor.2018.10.025

[16] Morgan, J. S., Howick, S., and Belton, V. (2017). "A toolkit of designs for mixing discrete event simulation and system dynamics". *European Journal of Operational Research*, **257(3)**: 907-918. https://doi.org/10.1016/j.ejor.2016.08.016

[17] Terziyan, V., and Kaikova, O. (2015). "The 'magic square': A roadmap towards emotional business intelligence". *Journal of Decision Systems*, **24(3)**: 255-272. https://doi.org/10.1080/12460125.2015.969592

[18] Talbi, E. G. (2013). "A unified taxonomy of hybrid metaheuristics with mathematical programming, constraint programming and machine learning". In: *Hybrid Metaheuristics* (pp. 3-76). Springer. https://doi.org/10.1007/978-3-642-30671-6_1

[19] Terziyan, V., and Puuronen, S. (2000). "Reasoning with multilevel contexts in semantic metanetworks". In: *Formal Aspects of Context* (pp. 107-126). Springer. https://doi.org/10.1007/978-94-015-9397-7_7

[20] Terziyan, V. (2005). “A Bayesian metanetwork”. *International Journal on Artificial Intelligence Tools*, **14(03)**: 371-384. https://doi.org/10.1142/S0218213005002156

[21] Savolainen, V. and Terziyan, V. (1999). "Metapetrinets for controlling complex and dynamic processes". *International Journal of Information and Management Sciences*, **10(1)**: 13-32. http://www.cs.jyu.fi/ai/vagan/papers/vt_vs_Petri.pdf

[22] Petri, C. A., and Reisig, W. (2008). ”Petri net”. *Scholarpedia*, **3(4)**: 6477. http://dx.doi.org/10.4249/scholarpedia.6477

[23] Latorre-Biel, J. I., Faulín, J., Juan, A. A., and Jiménez-Macías, E. (2018). “Petri net model of a smart factory in the frame of industry 4.0”. *IFAC-PapersOnLine*, **51(2)**: 266-271. https://doi.org/10.1016/j.ifacol.2018.03.046

[24] Grobelna, I., and Karatkevich, A. (2021). “Challenges in application of Petri nets in manufacturing systems”. *Electronics*, **10(18)**: 2305. https://doi.org/10.3390/electronics10182305

[25] Kari, J. (2005). “Theory of cellular automata: A survey.” *Theoretical Computer Science*, **334(1-3)**: 3-33. https://doi.org/10.1016/j.tcs.2004.11.021

[26] Sarkar, P. (2000). “A brief history of cellular automata”. *ACM Computing Surveys*, 32(1): 80-107. https://doi.org/10.1145/349194.349202

[27] Gardner, M. (1970). “Mathematical games”. *Scientific American*, **222(6)**: 132-140. https://www.jstor.org/stable/24925832

[28] Zoričak, M., Horváth, D., Gazda, V., and Hudec, O. (2021). “Spatial evolution of industries modelled by cellular automata”. *Journal of Business Research*, **129**: 580-588. https://doi.org/10.1016/j.jbusres.2019.12.043

[29] Mitra, A. (2021). “On the capabilities of cellular automata-based MapReduce model in industry 4.0”. *Journal of Industrial Information Integration*, **21**: 100195. https://doi.org/10.1016/j.jii.2020.100195

[30] Kazemi Kordestani, J., Razapoor Mirsaleh, M., Rezvanian, A., and Meybodi, M. R. (2021). “Cellular automata, learning automata, and cellular learning automata for optimization”. In: *Advances in Learning Automata and Intelligent Optimization* (pp. 75-125). Springer, Cham. https://doi.org/10.1007/978-3-030-76291-9_3

[31] LeCun, Y., Bengio, Y., and Hinton, G. (2015). “Deep learning”. *Nature*, **521(7553)**: 436-444. https://doi.org/10.1038/nature14539

[32] Sarker, I. H. (2021). “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions”. *SN Computer Science*, **2(6)**: 1-20. https://doi.org/10.1007/s42979-021-00815-1

[33] Saufi, S. R., Ahmad, Z. A. B., Leong, M. S., and Lim, M. H. (2019). “Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review”. *IEEE Access*, **7**: 122644-122662. https://doi.org/10.1109/ACCESS.2019.2938227

[34] Montavon, G., Samek, W., and Müller, K. R. (2018). “Methods for interpreting and understanding deep neural networks”. *Digital Signal Processing*, **73**: 1-15. https://doi.org/10.1016/j.dsp.2017.10.011

[35] Barragán, I., Seck-Tuoh, J. C., and Medina, J. (2013). “Relationship between Petri nets and cellular automata for the analysis of flexible manufacturing systems”. In: *Proceedings of Mexican International Conference on Artificial Intelligence* (pp. 338-349). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-37798-3_30

[36] Barragán, I., Tuoh, J. C. S., and Medina, J. (2015). “Petri nets representing the evolution of elementary cellular automata”. *IEEE Latin America Transactions*, **13(9)**: 3103-3112. https://doi.org/10.1109/TLA.2015.7350065

[37] Zaitsev, D. A., Ghaffari, P., and Sanchez, V. S. (2021). “Modeling Ebola virus dynamics by colored Petri nets”. In: *Frontiers in Artificial Intelligence and Applications* (Vol. 345, pp. 707-715). IOS Press. https://doi.org/10.3233/FAIA210464

[38] Vahidipour, S. M., Esnaashari, M., Rezvanian, A., and Meybodi, M. R. (2019). “GAPN-LA: A framework for solving graph problems using Petri nets and learning automata”. *Engineering Applications of Artificial Intelligence*, **77**: 255-267. https://doi.org/10.1016/j.engappai.2018.10.013

[39] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). ”Dropout: a simple way to prevent neural networks from overfitting”. *Journal of Machine Learning research*, **15(1)**: 1929-1958. https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

[40] Koivu, A., Kakko, J. P., Mäntyniemi, S., and Sairanen, M. (2022). ”Quality of randomness and node dropout regularization for fitting neural networks”. *Expert Systems with Applications*, **207**: 117938. https://doi.org/10.1016/j.eswa.2022.117938

[41] Herrmann, J. P., Atanasyan, A., Casser, F., and Tackenberg, S. (2023). “A Petri net architecture for real-time human activity recognition in work systems”. *Procedia Computer Science*, **217**: 1188-1199. https://doi.org/10.1016/j.procs.2022.12.317

[42] Hu, S., Zhang, J., and Li, Z. (2022). “Diagnosability enforcement in labeled Petri nets based on digital twins”. In: *Proceedings of the 8th International Conference on Control, Decision and Information Technologies* (Vol. 1, pp. 1279-1284). IEEE. https://doi.org/10.1109/CoDIT55151.2022.9804165

[43] Gehlot, V., Rokowski, P., Sloane, E. B., and Wickramasinghe, N. (2022). “Taxonomy, tools, and a framework for combining simulation models with AI/ML models”. In: *Proceedings of the Annual Modeling and Simulation Conference* (pp. 18-29). IEEE. https://doi.org/10.23919/ANNSIM55834.2022.9859494

[44] De, T., Giri, P., Mevawala, A., Nemani, R., and Deo, A. (2020). “Explainable AI: a hybrid approach to generate human-interpretable explanation for deep learning prediction”. *Procedia Computer Science*, **168**: 40-48. https://doi.org/10.1016/j.procs.2020.02.255

[45] van der Aalst, W. M. (2019). “Everything you always wanted to know about petri nets, but were afraid to ask”. In: *Business Process Management: Proceedings of the 17th International Conference* (pp. 3-9). Springer, Cham. https://doi.org/10.1007/978-3-030-26619-6_1

[46] Bhattacharjee, K., Naskar, N., Roy, S., and Das, S. (2020). “A survey of cellular automata: types, dynamics, non-uniformity and applications”. *Natural Computing*, **19**: 433-461. https://doi.org/10.1007/s11047-018-9696-8

[47] Leijnen, S., and Veen, F. V. (2020). “The neural network zoo”. *Proceedings,* **47(1)**: 9. MDPI. https://doi.org/10.3390/proceedings2020047009