Machine Learning for Cyber Physical Systems

Proceedings of the Conference

ML4CPS 2025

Fraunhofer IOSB

Helmut Schmidt Universität
Universität der Bundeswehr Hamburg

OPEN ACCESS

Oliver Niggemann - Jürgen Beyerer - Achim Kampker - Rui Yan Li - Görschwin Fey - Alois Kritl - Alexander Diedrich - Christian Kühnert

Editors

Machine Learning for Cyber Physical Systems

Proceedings of the Conference ML4CPS 2025

Editors

Prof. Jürgen Beyerer
Dr. Christian Kühnert
Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung
Karlsruhe, Germany

Prof. Oliver Niggemann
Dr. Alexander Diedrich
Helmut-Schmidt-Universität Hamburg
Hamburg, Germany

Prof. Görschwin Fey
Technische Universität Hamburg
Hamburg, Germany

Prof. Achim Kampker
Rui Yan Li
RWTH Aachen
Aachen, Germany

Alois Kritl
ARIC
Hamburg, Germany

https://doi.org/10.24405/20018

Preface

Cyber Physical Systems are characterized by their ability to adapt and learn from their environment. Applications include advanced condition monitoring, predictive maintenance, diagnosis tasks, and many other areas. All these applications have in common that Machine Learning and Artificial Intelligence are the key technologies. However, applying ML and AI to CPS poses challenges such as limited data, less understood algorithms, and the need for high algorithm reliability. These topics were a focal point at the 8th ML4CPS—Machine Learning for Cyber-Physical Systems Conference in Berlin, held from March 6th to 7th, where industry and research experts discussed current advancements and new developments.

Prof. Dr. Oliver Niggemann
Prof. Dr. Jürgen Beyerer
Prof. Dr. Görschwin Fey
Prof. Dr. Achim Kampker
Dr. Alexander Diedrich
Dr. Christian Kühnert
Alois Kritl

Contents

ASSESSING ROBUSTNESS IN DATA-DRIVEN MODELING OF CYBER-PHYSICAL SYSTEMS

Maximilian Schmidt, Swantje Plambeck, Goerschwin Fey
Institute of Embedded Systems, Computer Engineering
Hamburg University of Technology
Am Schwarzenberg-Campus 1, 21073 Hamburg, Germany
firstname.lastname@tuhh.de

ABSTRACT

Robustness is a key factor in the design and analysis of Cyber-Physical Systems (CPS), ensuring that systems function correctly even under perturbations. This paper investigates robustness within the data-driven modeling process, focusing on three core aspects: system robustness, model robustness, and learner robustness. We survey existing notions of robustness and propose unified formal definitions for each aspect, analyzing their interdependencies and their contributions to overall CPS performance. Additionally, we introduce a method for assessing the robustness of models generated by data-driven learning that is independent of both the model's internal representation and the learning paradigm used. Our approach leverages input perturbations combined with probabilistic analysis to evaluate how well a learned model handles input variations, particularly when formal guarantees are challenging to obtain. To demonstrate the practical application of our method, we conduct a case study on a temperature control system, using decision trees to model system behavior. By perturbing test data and analyzing the resulting model outputs, we identify non-robust regions near decision boundaries, thereby revealing potential vulnerabilities. The proposed framework offers valuable insights for enhancing system design and lays the groundwork for future research into robust machine learning models for CPS.

Keywords Robustness · Cyber-Physical Systems · Data-Driven Modeling

1 Introduction

Cyber-Physical Systems (CPSs) integrate digital components with physical processes to intelligently monitor and control the physical world [26]. The digital components collect data from the environment and use it to make control decisions that are executed by actuators interacting with the physical world [24]. CPSs are ubiquitous in modern society, with applications ranging from automotive systems [15] and industrial automation [27] to smart grids [14] and healthcare [11].

The design and analysis of such systems present several challenges, primarily due to the need for accurate models that capture both physical dynamics and control logic. Such models are essential for testing and validating CPS designs in simulated environments, thereby reducing risks associated with real-world testing, such as operational disruptions or safety hazards when exploring extreme scenarios. Models also play a key role in identifying design flaws early, guiding system specifications, and supporting the overall development process.

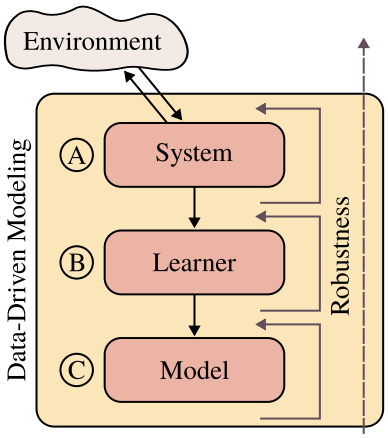


Figure 1: Workflow for data-driven modeling and the role of robustness analysis for the individual steps, as well as robustness as a property of the entire pipeline

Although systems can be precisely defined in theory, uncertainties in the physical environment—such as sensor inaccuracies, external disturbances, or even adversarial attacks—introduce deviations that can compromise system behavior. Given the inherent complexity of CPSs, exploring the entire behavioral space during the design phase is both computationally expensive and often impractical [17]. CPSs generate large amounts of data, which can be used to infer models through data-driven approaches. Observing the system during operation is thus more practical than analyzing it in full detail, provided that robust models are available to accurately predict system behavior.

For this paper, we adopt the data-driven modeling approach shown in Figure 1, where a *learner* is tasked with inferring a model of the system from collected data. We use the term *system* to represent a real CPS interacting with its environment. A *model* is a simplified representation of the system that abstracts its complexity to facilitate analysis, simulation, and prediction. We specifically focus on the robustness of the model generation pipeline. As CPSs increasingly operate in environments vulnerable to security attacks, robustness is essential to ensure resilience against adversarial threats [7, 3]. In the context of data-driven modeling of CPSs, robustness encompasses several dimensions, including *(A)* the system's resilience to input perturbations [7], *(B)* the learner's ability to generate functioning models despite variations in training data [28], and *(C)* the predictive accuracy of the model under unforeseen conditions [6]. Generally, we refer to robustness as the capacity of an entity to maintain stable performance in its environment even under conditions that were not anticipated during design. We note that *resilience* typically refers to the ability to adapt and recover from disturbances, whereas robustness focuses on maintaining stability in the presence of perturbations. Throughout this paper, we use the term *robustness* in the sense of stability and resistance, while acknowledging that resilience is a complementary property. This contrasts with the notion of *reliability*, which typically refers to the system's ability to perform its intended function without internal failure. Additionally, robustness is often associated with the ability to recover from environmental errors [5].

It is important to note that while robustness aims to ensure stability and resistance to perturbations, this often comes at the expense of sensitivity. A highly robust system or controller may react more slowly to important changes in its environment compared to a more sensitive design. Thus, designers must carefully balance robustness and sensitivity, ensuring that stability does not unduly hinder the systems responsiveness.

This paper explores various definitions and implications of robustness within CPS modeling. Rather than advocating for a single superior notion of robustness, we build on existing definitions by combining, abstracting, and formalizing them to offer a more comprehensive perspective on robustness in CPSs. Different application contexts require distinct formalizations of robustness due to the unique challenges and requirements each context presents [7].

The main contributions of this paper are:

- An overview of various notions of robustness found in the literature.
- A unified notion of robustness for data-driven modeling pipelines that addresses systems, models, and learners, along with a discussion of the implications.
- A data-centric method to analyze model robustness through input perturbation.
- An application of the proposed framework to a hybrid system, demonstrating its practical utility.

The remainder of this paper is structured as follows: section 2 provides an overview of related work on robustness in CPSs. In section 3 we introduce a unified notion of robustness for data-driven modeling, focusing on systems, models, and learners. Section 3.1 explores the robustness of a system, section 3.2 discusses the robustness of a model, and section 3.3 examines the robustness of a learner. Section 4 presents a data-centric framework for analyzing model robustness using perturbation. Finally, section 6 concludes the paper with a summary of the key findings and implications for future research.

2 Related Work

The term *robustness* is used in various contexts, and its definition varies with the application domain. Robustness commonly refers to a system's ability to tolerate perturbations and continue functioning effectively despite deviations from expected behavior, such as disturbances, uncertainties, or adversarial attacks. Literature on robustness in CPSs spans multiple disciplines, including control theory, machine learning, and formal methods. Table 1 provides an overview of related literature and summarizes how robustness is approached in different contexts.

In control theory, robustness is primarily associated with a system's ability to maintain stability and minimize performance degradation in the presence of perturbations, with an emphasis on graceful degradation under disturbances. CPSs typically consist of physical and digital components, where the physical component describes continuous behavior and the digital component describes discrete behavior. Analyzing robustness separately for these components follows naturally from the different types of disturbances affecting them. There are also approaches that combine control and digital logic, such as hybrid system robustness, which addresses robustness across both layers of CPSs. Formal methods provide a theoretical foundation for robustness analysis, often employing temporal logic to specify and verify robustness properties of CPSs.

From a machine learning perspective, robustness is often framed in terms of a model's resilience to adversarial samples or out-of-distribution inputs, ensuring that predictions remain reliable under such conditions. Another key focus in machine learning is optimizing data selection or transformation to enhance the robustness of both learners and models.

3 Robustness in Data-Driven Modeling

In the context of data-driven modeling for CPSs, we consider robustness at three distinct levels: the system, the model, and the learner. Each of these components plays a critical role in the overall performance and reliability of the modeling pipeline. We begin by examining system robustness, which concerns the system's ability to maintain stable behavior under external perturbations. Next, we analyze model robustness, focusing on the models capacity to accurately predict system behavior despite input variations. Finally, we address learner robustness, which relates to the learners ability to generate reliable models when trained on imperfect or perturbed data. By investigating robustness at each of these levels, we aim to provide a comprehensive understanding of the challenges and requirements for robust data-driven CPS modeling.

Reference	Area	Notion of Robustness
Tabuada et al. [25]	Control theory	Individual analysis of physical and digital components in CPSs
Rungger et al. [20]	Control theory	Deviation from nominal behavior proportional to the disturbance
Sontag [23]	Control theory	Input-state stability for robustness analysis
Bloem et al. [5]	Formal methods	Proportionality of environmental failures to system failures
Goebel et al. [10]	Formal methods	Homogeneous perturbations using pre-asymptotic stability
Castiglioni et al. [7]	Formal methods	Temporal logic for robustness analysis of CPSs
Hsieh [12]	Analysis	Resilience against resource failures in production systems
Cui et al. [9]	Analysis	Ratio of functioning nodes after an attack or error
Baker et al. [2]	Analysis	Risk and consequences of potential damage to the system
Zhang et al. [29]	Machine learning	Resilience against failures in digital components
Sehwag et al. [21]	Machine learning	Analyzing robustness with out-of-distribution samples
Ramoni et al. [19]	Machine learning	Robust learning under missing data
Bhagoji et al. [4]	Machine learning	Data transformations to enhance model robustness
Najafi et al. [16]	Machine learning	Role of unlabeled data in enhancing adversarial robustness
Odyurt et al. [17]	Monitoring	Variants of anomalies in traces of execution
Zhang et al. [30]	Monitoring	Cascading failures in complex systems and impact of failures
Shahrokni et al. [22]	Software	Software robustness
Hu et al. [13]	Design methods	Stability, security, and systematic design considerations

Table 1: Overview of related work on robustness in CPSs

3.1 Robustness of a System

Robustness of a system refers to its ability to maintain functionality in the presence of perturbations. A system

$$S : I \to O$$

defines a mapping from inputs to outputs, where I denotes the input space and O the output space. The systems behavior is characterized by this input-output relationship. Here, I and O may represent traces of inputs and outputs over time to capture time-dependent or stateful behavior. Deviations in system behavior arise when design assumptions are violated at runtime, which is inevitable due to partial knowledge about the system and its operating environment [20].

To formalize perturbations, we define a perturbation function $F : I \to I$ that modifies the input to the system, and we introduce a robustness predicate R that encapsulates the intended functionality of the system for assessing its robustness. Here, i denotes an original input, $F(i)$ is the perturbed input, $S(i)$ is the output corresponding to the original input, and $S(F(i))$ is the output under perturbation. The predicate $R \subset I \times I \times O \times O$ is defined to capture acceptable deviations in behavior; for instance, it may specify that the distance between $S(i)$ and $S(F(i))$ does not exceed a predefined threshold. A system is thus considered robust if

$$(i, F(i), S(i), S(F(i))) \in R \quad \forall i \in I.$$

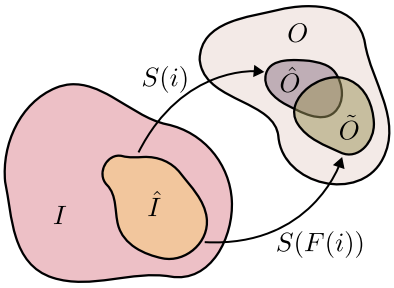


Figure 2: Input and output sets used to model robustness under input perturbations

The predicate may be restricted to a local subspace $\hat{I} \subset I$ to focus on specific regions of interest. Figure 2 illustrates the concept of robustness by showing how input perturbations affect the corresponding outputs.

A straightforward approach to assess robustness is to define a distance metric $d_o : O \times O \to \mathbb{R}$ between the intended output o and the perturbed output $\widetilde{o} = S(F(i))$. With a predefined threshold $\epsilon \in \mathbb{R}$, the robustness predicate R_ϵ is characterized by

$$d_o(\widetilde{o}, o) \leq \epsilon.$$

For this *bounded output stability* [1], the system is said to be ϵ-robust if R_ϵ holds for all perturbed outputs.

Another perspective on robustness is that the system's deviation from nominal behavior is proportional to the magnitude of the environmental disturbance [5, 20]. We denote this *proportional stability* as R_κ using a distance metric for both input and output spaces. We define an input space distance metric $d_i : I \times I \to \mathbb{R}$ similarly to d_o, and the system is considered robust if

$$d_o(\widetilde{o}, o) \leq \kappa \cdot d_i(\widetilde{i}, i),$$

where $\kappa \in \mathbb{R}$ is a proportionality constant, $\widetilde{i} = F(i)$, and $\widetilde{o} = S(\widetilde{i})$. This proportional stability is a specialization of the Lipschitz continuity property and is commonly assumed in CPS robustness analyses.

We call any perturbed input sample that causes a violation of the robustness predicate an *adversarial sample*. By this we transfer the notion of adversarial examples from the context of machine learning to CPS and define *adversarial robustness* as the magnitude of the disturbance required to transform any original sample into an adversarial one [6]. Specifically, adversarial robustness in a local input space $\hat{I}$ is quantified as the minimum distance between any adversarial sample and its corresponding original sample:

$$\min_{i \in \widetilde{I}} d_i(i, F(i)) \text{ s.t. } (i, F(i), S(i), S(F(i))) \notin R.$$

This metric provides a means to quantify the system's resilience to adversarial perturbations, indicating the level of disturbance required before its robustness fails.

3.2 Robustness of a Model

A model M is designed to replicate the behavior of a system S for the purposes of analysis and prediction, typically by abstracting some of the system's complexity. The model represents a simplified version of the system, capturing its essential input-output characteristics: $M : I \to O$.

The robustness of a model refers to its ability to accurately predict the system behavior despite perturbations. Assuming the system satisfies a robustness predicate R, the model—intended to faithfully represent the system behavior—should also satisfy R under similar perturbations. Thus, the model is subject to the same robustness definitions as the system.

3.2.1 Models as Surrogates

To quantify the difference between the model output and the system output, we define a loss function $L : O \times O \to \mathbb{R}$ that enables the optimization of an accurate model representation of the system behavior. An accurate model is one that minimizes this loss function. Although system robustness cannot be directly inferred from model robustness, a sufficiently accurate model can serve as a surrogate, providing valuable insights into the robustness properties of the system.

This strategy offers two key advantages. First, by identifying regions where the model fails to meet robustness criteria, we can cross-reference these findings with the system to determine whether the issues stem from limitations in the model or reflect inherent weaknesses in the system. This feedback loop between model and system helps to clarify whether non-robust behavior is intrinsic to the system design or simply a byproduct of the modeling process.

Second, identifying non-robust regions via the model yields actionable insights for improving the system design, leading to enhanced overall performance. In this way, the model serves not only as a diagnostic tool but also as guidance for system improvements.

3.2.2 Limitation

If the system fails to satisfy a robustness predicate, then even an accurately constructed model cannot be expected to meet that predicate. This implies that the robustness of a model is inherently limited by the robustness of the system it represents. Therefore, while a model may be validated under nominal conditions, if the underlying system exhibits non-robust behavior in certain scenarios, the models predictions under similar conditions will likewise lack robustness.

3.3 Robustness of a Learner

A learner is an algorithm tasked with inferring a model from training data. The performance of the model is evaluated based on its ability to accurately predict system behavior. The optimal model is obtained by minimizing the loss function defined in section 3.2.1:

$$\min_{M} \sum_{i \in I} L(M(i), S(i)) \quad \forall i \in I.$$

The robustness of a learner $\mathcal{L}$ refers to its ability to produce an accurate model despite perturbations in the training data. Since the predictive performance of a model depends on the quality of the training data, learner robustness measures its capacity to adapt to variations in that data. Formally, we denote the learning process as $\mathcal{L}(D) = M$, where $D \subset I \times O$ is the training set of input-output samples and M is the model inferred by the learner. The optimal learner $\mathcal{L}^*$ is one that infers a model that perfectly captures the system behavior, i.e.,

$$\mathcal{L}^*(D)(i) = S(i) \quad \forall i \in I.$$

To assess learner robustness, we define a perturbation function F on the training set. We then compare the performance of the model trained on the original training set with that of the model trained on the perturbed training set. The formal notation for this comparison, analogous to the system robustness definition (cf. section 3.1), is omitted here for brevity.

Training samples that mislead the learner into generating suboptimal models are termed adversarial training samples. The learner's resilience against these adversarial samples is a key aspect of its robustness. Mathematical frameworks can formally guarantee robustness properties, as demonstrated by generalization results for neural networks and splitting criteria for decision trees [8].

Robust learning is closely related to learning under missing or limited observation data. Perturbations in training data involve more than merely adding or omitting samples; for CPSs, common issues include noise and non-equidistant sampling rates that distort the true distribution and lead to inaccurate representations. Furthermore, if the system generating the training data is not robust, the learner cannot necessarily compensate for perturbed data, as such perturbations may lead to inconsistent learning outcomes.

4 Analyzing Robustness

Robustness has long been a focus in various fields: system robustness is extensively studied in control theory, while learner robustness is commonly investigated in machine learning, particularly in the context of adversarial examples. Having discussed the three core components of the data-driven modeling pipeline—systems, models, and learners—we now shift our focus to model robustness.

4.1 Methodology

The method for evaluating robustness largely depends on the type of model being analyzed. For black-box models, robustness assessment typically involves perturbing the inputs and observing the resulting outputs to detect any deviations. This input-output approach does not require insight into the internal structure of the model. In contrast, white-box models permit structural analysis, which enables direct inference of robustness properties from the model internals.

In addition to explicit formalization, a data-driven and learning-based approach can be applied to robustness analysis. Instead of relying solely on formal methods or direct structural analysis, this approach uses data to predict potential robustness violations. By analyzing input-output pairs and their perturbations, models can be trained to detect patterns indicative of robustness issues. This predictive method is especially useful when formal robustness guarantees are challenging to obtain in highly complex models or systems.

4.2 Input Perturbation

We propose a black-box method for assessing robustness across the entire data-driven modeling pipeline, with a particular focus on the learned model. Our approach evaluates how the model responds to variations in the input space by systematically applying perturbations to the test data. This enables a comprehensive analysis of the models ability to generalize beyond its training data and remain robust under input variations.

Analyzing model robustness through sample testing is analogous to evaluating a model on a test set where the test set comprises perturbed samples. This method is particularly useful for complex models, such as (deep) neural networks, where deriving robustness properties analytically is challenging.

4.3 Probabilistic Robustness Analysis

To perform a probabilistic robustness analysis, we begin by training a model M on a dataset D that captures the behavior of the system S. By perturbing each input sample from the dataset with a randomly drawn perturbation F, we augment the test dataset and conduct a probabilistic assessment of the models robustness.

We choose a distance metric $d_o : O \times O \to \mathbb{R}$ to quantify the difference between the model output on the original input and the output on the perturbed input. The choice of perturbation function F and distance metric d_o must be based on the model type and the application domain. For example, the perturbation function may involve adding Gaussian noise to the input, while the distance metric d_o may be the L_0, L_2, or L_∞ distance.

It should be noted that the probabilistic robustness analysis does not provide formal guarantees. The statistical measures obtained are sensitive to the chosen perturbation function and distance metric. The coverage of the perturbation space may be limited, and thus the conclusions drawn from this analysis should be interpreted with caution. Future work will explore methods to enhance the formal coverage of the analysis.

This approach provides a flexible and scalable method for evaluating robustness, particularly for black-box models where formal analysis is impractical. By systematically perturbing inputs and assessing the impact on model outputs, we can determine how well the model maintains its functionality under diverse conditions. Due to its probabilistic nature, this approach quantifies

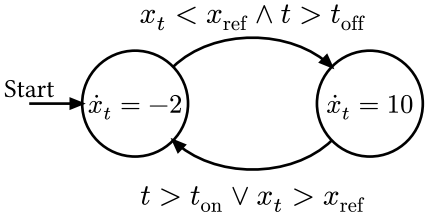


Figure 3: Automaton of the temperature control system

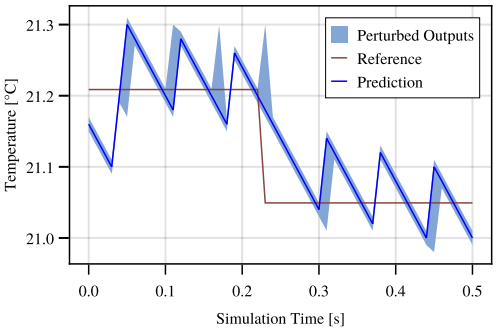


Figure 4: Predictions of the thermostat model with a probabilistic robustness analysis showing outputs under perturbations

robustness in terms of statistical measures but does not provide formal guarantees. We leave an exploration of the quality and coverage of this probabilistic analysis for future work.

5 Case Study

In this case study, we apply our probabilistic robustness analysis method to a simple CPS: a temperature control system. We use this system to illustrate how input space perturbations can reveal insights into the robustness of a model trained to predict system behavior.

5.1 Temperature Control System

The temperature control system consists of two main components: the boiler and the thermostat. The thermostat monitors the current temperature and compares it with a reference temperature. Based on this comparison, it switches the boiler on or off. The hybrid system, illustrated in Figure 3, operates in two modes: on and off.

To prevent overheating, the system enforces a predefined maximum duration for which the boiler can remain on, after which it automatically switches off. Additionally, a cooldown mechanism ensures that the system cannot immediately switch back to the on mode after being turned off, thereby introducing a delay before reheating is possible.

The system takes one input: the reference temperature $x_{\text{ref}} \in \mathbb{R}$. The internal state of the system is represented by the current temperature $x_t \in \mathbb{R}$. The rate of temperature change is governed by its first derivative: $\dot{x}_t = -2$ in the off mode and $\dot{x}_t = 10$ in the on mode.

5.2 Robustness Analysis

To analyze the robustness of the system, we first simulate the temperature control system for 100,000 samples, thereby generating a dataset for training. The dataset encodes the system input as a trace consisting of the current temperature x_t^n and the reference temperature x_{ref}^n over multiple time steps, while the system output is the next temperature x_t^{n+1}:

$$i = [x_t^0, x_{\text{ref}}^0, \ldots, x_t^n, x_{\text{ref}}^n, x_{\text{ref}}^{n+1}] \quad o = x_t^{n+1}.$$

We split the dataset into training and test sets and train a decision tree model to predict the next temperature based on the input trace. Decision trees are well-suited for learning the behavior of hybrid systems, as noted in prior work [18]. The trained model achieves a mean absolute error of 0.001 on the test set, indicating high accuracy in predicting system behavior.

Next, we assess the robustness of the model by applying perturbations to the last input reference. Specifically, we perturb the input by adding a disturbance $f \in \mathcal{U}_{[-\delta,\delta]}$, where $\mathcal{U}$ denotes a uniform distribution, i.e., $F(i) = i + f$. The model robustness is measured using the absolute

deviation between the models output on the perturbed input $\tilde{o} = M(F(i))$ and its output on the original input $o = M(i)$, given by $d_o(\tilde{o}, o) = |\tilde{o} - o|$. Each input sample from the test set is perturbed 100 times using randomly drawn disturbances, and the model behavior is observed.

Figure 4 illustrates the simulation results, showing the predicted model output (in blue) along with the disturbed outputs represented as an area around the prediction. The robustness analysis reveals non-robust regions in the output, particularly near the decision boundaries between the systems on and off states. These regions exhibit greater sensitivity to input perturbations, highlighting areas where the model predictions are less robust.

This case study demonstrates the utility of probabilistic robustness analysis in identifying potential vulnerabilities in CPS models. By systematically perturbing the input space, we gain a deeper understanding of the model's response to variations, which can inform improvements in system design or guide further model refinement. If access to the actual system is available, these insights can also be used to conduct targeted testing and analysis.

In addition to detecting robustness violations, the model can serve diagnostic purposes, such as identifying faulty samples and classifying the nature of the faults. This extends the robustness analysis beyond mere detection, offering a deeper understanding of the perturbations or failures that lead to robustness violations.

6 Conclusion

Robustness is a critical attribute in the design and analysis of CPSs, ensuring that systems, models, and learners can withstand and adapt to disturbances, uncertainties, and variations in their operational environment. This paper has explored the definitions and implications of robustness for data-driven modeling from three perspectives: systems, models, and learners. Overall, robustness is a multifaceted concept that is pivotal in ensuring reliability, security, and performance. Our exploration underscores the importance of robust design and analysis in building resilient and trustworthy CPSs that can operate effectively despite inevitable disturbances and uncertainties. Future work will continue to refine robustness metrics and develop methods to enhance the robustness of systems, models, and learners. Furthermore, the probabilistic robustness analysis presented here, while flexible, lacks formal guarantees and its coverage is inherently limited. The discussion on learner robustness, in particular, would benefit from further investigation into strategies such as adversarial training and robust loss formulations. Finally, additional work is needed to more precisely formalize the robustness predicates and ensure that all components of the modeling pipeline are thoroughly evaluated.

Acknowledgments

This work is funded by BMBF project AGenC no. 01IS22047A.

References

[1] B. D. O. Anderson and J. B. Moore. New Results in Linear System Stability. *SIAM Journal on Control*, 7(3):398–414, Aug. 1969.

[2] J. W. Baker, M. Schubert, and M. H. Faber. On the assessment of robustness. *Structural Safety*, 30(3):253–267, May 2008.

[3] A. Banerjee, K. K. Venkatasubramanian, T. Mukherjee, and S. K. S. Gupta. Ensuring Safety, Security, and Sustainability of Mission-Critical Cyber–Physical Systems. *Proceedings of the IEEE*, 100(1):283–299, Jan. 2012.

[4] A. N. Bhagoji, D. Cullina, C. Sitawarin, and P. Mittal. Enhancing robustness of machine learning systems via data transformations. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5, Princeton, NJ, Mar. 2018. IEEE.

[5] R. Bloem, K. Greimel, T. A. Henzinger, and B. Jobstmann. Synthesizing robust systems. pages 85–92, Dec. 2009.

[6] N. Carlini and D. Wagner. Towards Evaluating the Robustness of Neural Networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, San Jose, CA, USA, May 2017. IEEE.

[7] V. Castiglioni, M. Loreti, and S. Tini. RobTL: A Temporal Logic for the Robustness of Cyber-Physical Systems, Dec. 2022.

[8] H. Chen, H. Zhang, S. Si, Y. Li, D. Boning, and C.-J. Hsieh. Robustness Verification of Tree-based Models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[9] P. Cui, P. Zhu, P. Xun, and Z. Xia. Enhance the robustness of cyber-physical systems by adding interdependency. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 203–203, July 2017.

[10] R. Goebel and A. R. Teel. Preasymptotic Stability and Homogeneous Approximations of Hybrid Dynamical Systems. *SIAM Review*, 52(1):87–109, Jan. 2010.

[11] S. A. Haque, S. M. Aziz, and M. Rahman. Review of Cyber-Physical System in Healthcare. *International Journal of Distributed Sensor Networks*, 10(4):217415, Apr. 2014.

[12] F.-S. Hsieh. Robustness Analysis of Cyber-Physical systems based on Discrete Timed Cyber-Physical Models. In *2021 IEEE World AI IoT Congress (AIIoT)*, pages 0250–0254, May 2021.

[13] F. Hu, Y. Lu, A. V. Vasilakos, Q. Hao, R. Ma, Y. Patil, T. Zhang, J. Lu, X. Li, and N. N. Xiong. Robust Cyber–Physical Systems: Concept, models, and implementation. *Future Generation Computer Systems*, 56:449–475, Mar. 2016.

[14] S. Karnouskos. Cyber-Physical Systems in the SmartGrid. In *2011 9th IEEE International Conference on Industrial Informatics*, pages 20–23, Lisbon, Portugal, July 2011. IEEE.

[15] M. Lukasiewycz, S. Steinhorst, F. Sagstetter, W. Chang, P. Waszecki, M. Kauer, and S. Chakraborty. Cyber-Physical Systems Design for Electric Vehicles. In *2012 15th Euromicro Conference on Digital System Design*, pages 477–484, Cesme, Izmir, Turkey, Sept. 2012. IEEE.

[16] A. Najafi, S.-i. Maeda, M. Koyama, and T. Miyato. Robustness to Adversarial Perturbations in Learning from Incomplete Data. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[17] U. Odyurt, A. D. Pimentel, and I. Gonzalez Alonso. Improving the robustness of industrial Cyber–Physical Systems through machine learning-based performance anomaly identification. *Journal of Systems Architecture*, 131:102716, Oct. 2022.

[18] S. Plambeck and G. Fey. Decision Tree Models of Continuous Systems. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sept. 2022.

[19] M. Ramoni and P. Sebastiani. Robust Learning with Missing Data. *Machine Learning*, 45(2):147–170, 2001.

[20] M. Rungger and P. Tabuada. A Notion of Robustness for Cyber-Physical Systems. *IEEE Transactions on Automatic Control*, 61(8):2108–2123, Aug. 2016.

[21] V. Sehwag, A. N. Bhagoji, L. Song, C. Sitawarin, D. Cullina, M. Chiang, and P. Mittal. Analyzing the Robustness of Open-World Machine Learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 105–116, London United Kingdom, Nov. 2019. ACM.

[22] A. Shahrokni and R. Feldt. A systematic review of software robustness. *Information and Software Technology*, 55(1):1–17, Jan. 2013.

[23] E. D. Sontag. Input to State Stability: Basic Concepts and Results. In A. A. Agrachev, A. S. Morse, E. D. Sontag, H. J. Sussmann, V. I. Utkin, P. Nistri, and G. Stefani, editors, *Nonlinear and Optimal Control Theory: Lectures given at the C.I.M.E. Summer School Held in Cetraro, Italy June 19–29, 2004*, Lecture Notes in Mathematics, pages 163–220. Springer, Berlin, Heidelberg, 2008.

[24] M. Szczodrak, Y. Yang, D. Cavalcanti, and L. P. Carloni. An open framework to deploy heterogeneous wireless testbeds for Cyber-Physical Systems. In *2013 8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, pages 215–224, Porto, June 2013. IEEE.

[25] P. Tabuada, S. Y. Caliskan, M. Rungger, and R. Majumdar. Towards Robustness for Cyber-Physical Systems. *IEEE Transactions on Automatic Control*, 59(12):3151–3163, Dec. 2014.

[26] W. M. Taha, A.-E. M. Taha, and J. Thunberg. *Cyber-Physical Systems: A Model-Based Approach*. Springer International Publishing, Cham, 2021.

[27] K. Thramboulidis. A cyber–physical system-based approach for industrial automation systems. *Computers in Industry*, 72:92–102, Sept. 2015.

[28] J. Uesato, J.-B. Alayrac, P.-S. Huang, R. Stanforth, A. Fawzi, and P. Kohli. Are Labels Required for Improving Adversarial Robustness?, 2019.

[29] J. J. Zhang, K. Liu, F. Khalid, M. A. Hanif, S. Rehman, T. Theocharides, A. Artussi, M. Shafique, and S. Garg. Building Robust Machine Learning Systems: Current Progress, Research Challenges, and Opportunities. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 1–4, New York, NY, USA, June 2019. Association for Computing Machinery.

[30] X. Zhang, H. Ma, and C. K. Tse. Assessing the Robustness of Cyber-Physical Power Systems by Considering Wide-Area Protection Functions. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 12(1):107–114, Mar. 2022.

AN ILLUMINATION BASED BACKDOOR ATTACK AGAINST CRACK DETECTION SYSTEMS IN LASER BEAM WELDING

Wenjie Huo[1], **Lennart Schmies**[2], **Andrey Gumenyuk**[2], **Michael Rethmeier**[2], **and Katinka Wolter**[1]
[1]Mathematics and Computer Science, Free University Berlin, Berlin, Germany
[2]BAM - Bundesanstalt für Materialforschung und -prüfung, Fachbereich Schweißtechnische Fertigungsverfahren, Berlin, Germany

ABSTRACT

Deep neural networks (DNNs) have been wildly used in engineering and have achieved state-of-the-art performance in prediction and measurement tasks. A solidification crack is a serious fault during laser beam welding and it has been proven to be successfully detected using DNNs. Recently, research on the security of DNNs is receiving increasing attention because it is necessary to explore the reliability of DNNs to avoid potential security risks. The backdoor attack is a serious threat, where attackers aim to inject an inconspicuous pattern referred to as trigger into a small portion of training data, resulting in incorrect predictions in the reference phase whenever the input contains the trigger. In this work, we first generate experimental data containing actual cracks in the welding laboratory for training a crack detection model. Then, targeting this scenario, we design a new type of backdoor attack to induce the model to predict the crack as a normal state. Considering the stealthiness of the attack, a common phenomenon during the welding process, illumination, is used as the backdoor trigger. Experimental results demonstrate that the proposed method can successfully attack the crack detection system and achieve over 90% attack success rate on the test set.

Keywords welding crack detection · backdoor attack · deep neural networks · system security

1 Introduction

In the past few years, deep neural networks (DNNs) have achieved great success in many computer vision tasks, and then have been widely applied in the industrial field to solve prediction and measurement problems. For example, the development of welding image recognition has enabled automatic quality inspection, which is beneficial for adjusting the welding process and improving the welding quality. Zhang et al. [1] design an 11 layers convolutional neural network (CNN) to identify three weld penetration defects. A real-time weld defect detection system is proposed in [2], which achieves a classification accuracy of over 99% on a real-world production line. To relieve of the small data problem, a semi-supervised transfer learning based network is proposed for identifying and quantifying weld defects [3]. In addition, in order to learn time sequence information, [4] builds and compares two models based on LSTM and 3DCNN, respectively, to extract spatiotemporal features, thereby detecting the occurrence of solidification cracks earlier.

The above DNN based applications are highly sensitive to security, as the failure to detect defects in time may result in unexpected costs. However, recent studies have revealed that DNNs are vulnerable to various attacks, leading to erroneous decisions and security concerns [5]. One well-known attack is the backdoor attack [6]. Attackers first inject a dedicated designed trigger into a small set of training data, and then poison the model by strongly associating the trigger with a pre-defined label commonly referred to as the target label. The victim model still behave normally on benign data, which means it is difficult detect on normal inputs. But it will predict the input as the target class as expected by the attacker when the trigger is pasted onto the image.

Existing backdoor attacks can be divided into two categories according to whether they change the label of the poisoned data: 1) poisoned-label attacks and 2) clean-label attacks. The first poisoned-label attack is called BadNet [6], where several pixels at the bottom of the image are tampered with, and then these images are mislabelled as the specific target class. It proves the feasibility of the backdoor attack and after this pioneering work, researchers are devoted to develop invisible triggers to avoid the exposure of poisoned data. Some representative works such as Blend [7], where the pixels of the original image are blended with the trigger of the same size, make it very difficult to detect the poisoned images with the naked eye. WaNet [8] generates a fixed warping field and then the image pixel locations are modified according to the warping field. Another backdoor method, ISSBA [9], trains an encoder-decoder network to embed a string into images, making these images contain invisible string information. Clean-label attacks only corrupt samples of the target class, making the attack more stealthy. The first clean-label attack LC [10] generates small changes to the target class inputs to make them harder to classify, resulting in the model relying on the trigger to classify the input into the target class. Other clean-label attacks, such as SIG [11] design a sinusoidal signal with given frequency as the trigger and Refool [12] exploits physical reflections to generate the trigger. Due to limited space, we refer the interested reader to more interesting backdoor attacks in the surveys [13, 14].

Inspired by existing works, we consider attacking the crack detection model to make it fail to predict the occurrence of solidification cracks during the welding process. The significance of this work lies in exposing the potential risks of machine learning models and providing a reminder to prevent such attacks. However, the triggers of existing backdoor attacks are either artificial, which will be very conspicuous in welding scenarios and can be easily detected by the human eye. Or they are not common in welding scenarios, such as using the natural phenomena shadow [15], projection [16] and reflection [12] to poison samples. In this paper, we design a new type of trigger to achieve a stealthy attack. The trigger is generated by a very common physical phenomenon, illumination, which is highly likely to occur during the welding process. We use experimental data which are collected in the welding lab at BAM. The solidification cracks are generated through the Controlled Tensile Weldability test (CTW test), which forces the specimen to crack by external stress [17]. Finally, we conduct experiments on this dataset to demonstrate that the illumination trigger can be successfully planted into DNN models.

Through this study, we explore the scenarios of model failure, which may not only be attributed to malicious attackers, but also to unexpected accidents during the welding process. We expect that this research can help further improve security and establish a trustworthy detection network in the future.

2 Threat model

Firstly, we give a specific definition of classification models and backdoor attacks. In the deep learning based welding crack detection system, the classifier f_θ is trained on a benign dataset $\mathcal{D}_b = \{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^{N}$, where inputs $\mathcal{X} \subset \mathbb{R}^d$ are continuous welding frames and $\mathcal{Y} = \{0, 1\}$ are the corresponding labels, representing no crack (negative) and crack (positive). The goal of f_θ is to make the predicted label $\hat{y} \triangleq \arg\max f_\theta(x_i)$ the same as the ground truth $y \in \mathcal{Y}$, where $f_\theta(x_i)$ is the confidence of two classes. The parameters θ are determined by minimizing the cross-entropy loss function $\mathcal{L}_{CE}$.

$$\theta = \arg\min \sum_{i=1}^{|D_b|} \mathcal{L}_{CE}(\mathcal{F}(x_i), y_i) \tag{1}$$

In the backdoor attack, attackers typically do not know the CNN model and cannot control the training process, but can access the training data. They need to achieve two goals. Firstly, users have a small amount of data to validate the accuracy of the model. Thus, attackers should ensure the accuracy on a benign dataset to avoid suspicion. Secondly, for inputs that contain the trigger, the model should predict them as the category specified by the attacker. The specified class is negative for crack detection models. These two goals can be expressed as:

$$\begin{cases} \arg\max f_\theta(x_i) = y_i \\ \arg\max f_\theta(x_i') = 0 \end{cases} \tag{2}$$

where the x_i' represent the poisoned input.

3 Implementation

In this section, we will describe the implementation of the backdoor attack. Firstly, we generate a realistic dataset through the CTW test. After that, we introduce the achievement of illumination simulation and the proposed illumination based backdoor attack during the welding process.

3.1 Data acquisition

The welding experiments were carried out on sheets of austenitic Cr-Ni steel AISI 304 (1.4301) with a thickness of 1 mm (120 mm x 600 mm). The experimental setup is shown in Figure 1. The welding parameters applied were 1 kW laser power and a constant welding speed of 1.0 m/min at a focus position of +5 mm. Argon with a flow rate of 20 l/min was used as shielding gas. All welds were bead-on-plate welds with the seam length of 100 mm.

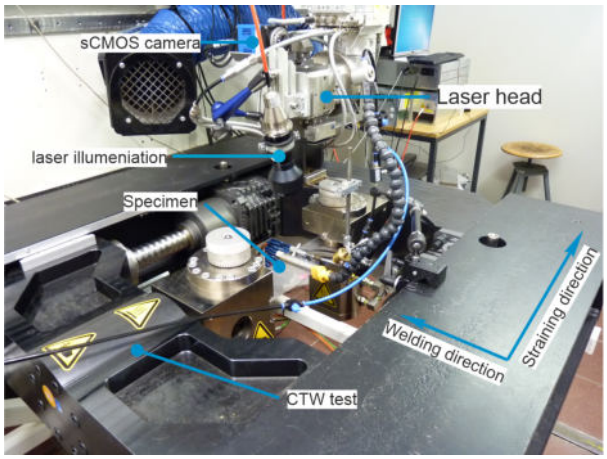


Figure 1: The experimental setup of the laser beam welding.

The defective samples containing solidification cracks were generated by an external load during the welding process. Figure 2 shows the CTW test procedure, which aims to induce cracking in the specimen by applying a plane tensile strain perpendicular to the welding direction at the predefined strain ϵ_{max} and strain rate $\dot{\epsilon}$. In these experiments, the specimens were subjected to different strain values from 0.02 to 0.035 and strain rates from 0.02 s^{-1} to 0.08 s^{-1}. In the beginning, no external strain was applied and the weld seam length reached approximately 20mm. Then an external strain with the predefined settings was applied to the specimen. Finally, the strain reaches the maximum value and remains constant until the welding was completed.

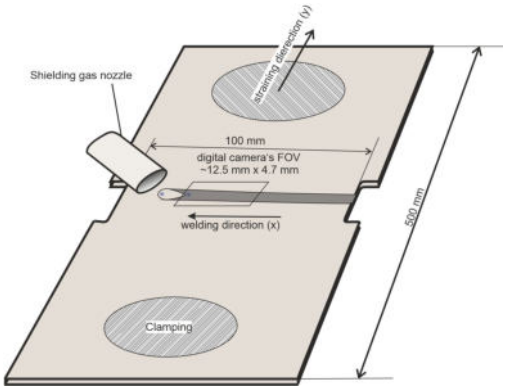


Figure 2: The CTW test procedure during laser beam welding.

The whole process of the laser beam welding was recorded with the PCO edge 5.5 camera, which has a sCMOS sensor as shown in Figure 1. To capture effective images, a diode laser (compact diode laser) from Dilas with a wavelength of 808 nm and a maximum power of 100 W was used as the illumination source. The recording rate used was 778 fps and the images were stored in tif format with a resolution of 800×130 pixels. Table 1 lists all experiments performed under different parameters. When the strain and strain rate are at low values, the welding status is normal, labeled as negative. And when they exceed the critical value, solidification cracks will form, which are labeled as positive.

Table 1: Strain and strain rate settings for CTW test.

	$\epsilon_{max} = 0.035$	$\epsilon_{max} = 0.03$	$\epsilon_{max} = 0.025$	$\epsilon_{max} = 0.02$
$\dot{\epsilon} = 0.02s^{-1}$	positive	positive	negative	negative
$\dot{\epsilon} = 0.04s^{-1}$	-	positive	negative	negative
$\dot{\epsilon} = 0.06s^{-1}$	-	-	positive	negative
$\dot{\epsilon} = 0.08s^{-1}$	-	-	positive	negative

3.2 Illumination simulation

During the laser beam welding, some environmental factors may affect the quality of collected images. For example, an external light source accidentally shine on the metal sheet or camera. To simulate illumination during the welding process, three important physical parameters should be considered: location, radius and intensity.

- Location $\mathcal{L}$: For the location of illumination, we use coordinate $\mathcal{L}(C_x, C_y)$ to represent the center point of the lighting which can be projected at any position on the welding specimen.
- Radius $\mathcal{R}$: $\mathcal{R}$ determines the region of illumination. When $\mathcal{R}$ is too small, it is hard to inject the backdoor trigger into the model. And when it is too large, the lighting will be easily noticed by humans. Thus, on the premise of successful attack, $\mathcal{R}$ should be as small as possible to ensure the stealthiness of the attack.
- Intensity $\mathcal{I}$: $\mathcal{I}$ is the intensity of light. Generally, the intensity of the center point is the strongest and gradually weakens towards the surroundings.

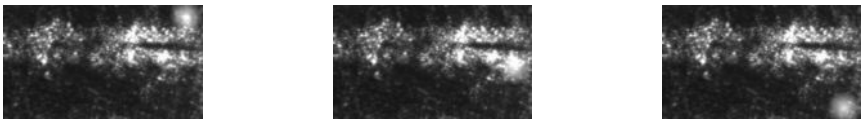
Based on the above parameters, we can generate the illumination trigger $\mathcal{T}[i, j]$ by

$$\begin{aligned} \mathcal{D}[i,j] &= \sqrt{(i-C_x)^2+(j-C_y)^2} \\ \mathcal{T}[i,j] &= \mathcal{I} \times (1-\mathcal{D}[i,j]/\mathcal{R}) \end{aligned} \tag{3}$$

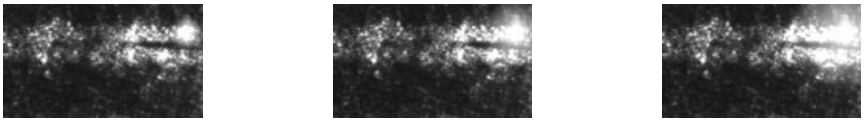
where $\mathcal{D}[i,j]$ is the distance between pixel (i,j) from the center of illumination $\mathcal{L}(C_x,C_y)$.

After simulating illumination, given a benign welding image x, the poisoned image x' is generated by

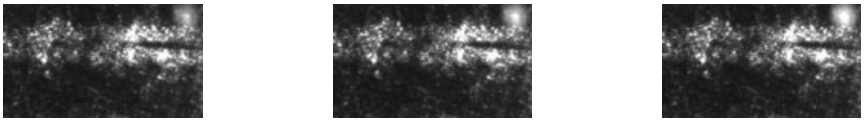
$$x'[i,j] = \begin{cases} min(255, \mathcal{T}[i,j]+x[i,j]) & \text{if } \mathcal{D}[i,j] < \mathcal{R} \\ x[i,j] & \text{otherwise} \end{cases} \tag{4}$$



(a) Location $\mathcal{L}$



(b) Radius $\mathcal{R}$



(c) Intensity $\mathcal{I}$

Figure 3: Comparison of poisoned samples by illumination with different locations, radii and intensities.

Figure 3 illustrates some instances of simulated illumination on welding dataset, including different locations, radii raging from 10 to 30, and intensities ranging from 100 to 200.

3.3 Illumination based backdoor attack

Figure 4 shows the process of injecting the trigger into the model. The training dataset consists of original benign samples and a small amount of samples generated in the previous section. There are two approaches to achieve the attack according to whether to change the label, namely poisoned-label attack or clean-label attack.

Poisoned-label attack. In the poisoned-label attack, a small number of samples are randomly collected. Then, these selected samples are poisoned by adding some simulated lighting and their ground truth labels are set to negative. The attack is achieved by learning the malicious mapping between illumination and negative class, therefore any frames with illumination will be predicted as negative. The poisoned samples can be represented as $\mathcal{D}_p = \{(x'_i, y_i = 0)\}_{i=1}^{N\times\gamma}$, and $\gamma = \frac{|\mathcal{D}_p|}{|\mathcal{D}_b+\mathcal{D}_p|}$ is the poisoning rate.

Clean-label attack. The poisoned-label attack contains clearly mislabeled data, thus poisoned samples may be detected under human inspection even though the lighting is not strong. To prevent the poisoned samples from being suspected, it is best to keep the input consistent with its label.

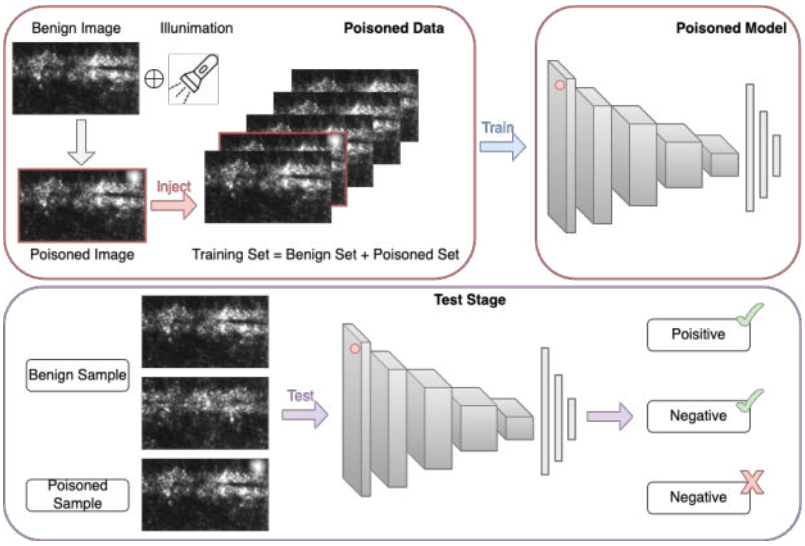


Figure 4: The training and test procedures of the illumination backdoor attack.

To achieve this goal, we try to perform illumination attack on a part of negative samples and induce the model to believe that illumination is associated with the negative category without modifying the labels. However, this does not lead to a high attack success rate. The reason could be that the model can classify correctly based on other obvious features rather than lighting features, so that it is difficult to convince the model to rely on illumination to classify inputs as the negative class.

To solve this problem, we change the contrast of the poisoned samples to make it more difficult for the model to learn normal features, thus relying more on lighting features for classification. Figure 5 shows the comparison before and after changing the contrast of one frame, which will not be suspicious under human inspection. In this way, the relationship between lighting and the negative class can be successfully established. In the clean-label attack, poisoned samples $\mathcal{D}_p = \{(x_i', y_i)|y_i = 0\}_{i=1}^{N \times R_p}$ all belong to the negative class and their labels do not need to be changed. Therefore the attack is more stealthy but the attack success rate is not as high as that of the poisoned-label attack.

(a) Original frame (b) Enhanced contrast frame

Figure 5: Frames before and after changing contrast.

4 Experiments

In this section, we first provide the experimental settings and attack goals. Then we evaluate the effectiveness of the illumination attack and compare the effects under different parameters.

4.1 Experimental setup and objectives

Classification model. As the welding dataset only contains two categories and has simple features, we do not select very complex network structures. The baseline network used in

the experiment is as follows: we use 5 convolutional layers and each layer is followed by a max pooling layer; there are 2 fully connected layers with 256 and 2 output neurons. For all convolutional layers, the kernel size is 3×3 with stride 1, and for pooling layers it is 2×2 with stride 1. The hyper-parameters of each layer are listed in Table 2.

Table 2: Architecture of the Model

Layer	Operator	Channels	Kernel Size	Output Size
L1	Conv1	16	3	$16 \times 64 \times 128$
	Pool1		2	$16 \times 32 \times 64$
L2	Conv2	32	3	$32 \times 32 \times 64$
	Pool2		2	$32 \times 16 \times 32$
L3	Conv3	48	3	$64 \times 16 \times 32$
	Pool3		2	$64 \times 8 \times 16$
L4	Conv4	56	3	$128 \times 8 \times 16$
	Pool4		2	$128 \times 4 \times 8$
L5	Conv5	64	3	$196 \times 4 \times 8$
	Pool5		2	$196 \times 2 \times 4$
L6	FC1	-	-	1×64
L7	FC2	-	-	1×2

Model training. The crack detection network is trained on 10 epochs, the batch size is 128. The optimizer uses Stochastic Gradient Descent (SGD) with learning rate 0.01, momentum 0.9 and weight decay 0.0005. Under this setting, the accuracy of the baseline network reaches 99.26% without attacks.

Attack setting. In our backdoor attack, the target label is $y_t = 0$, thus for the clean-label attack, the illumination trigger is injected into negative class videos. For the poisoned-label attack, one positive video is poisoned and its label is modified. Table 3 lists the data poisoning setting for two types of attacks. The total frame count of a welding video is nearly 4000 frames, while the number of frames from crack formation to disappearance is only about 200 frames. The poisoning rate is calculated based on the number of frames. It can be seen that although clean-label attacks are more stealthy, its disadvantage is that a larger proportion of training samples need to be corrupted to achieve the same goal. In addition to the parameters in the table, we also investigate the results of different poisoning rates in the end.

Table 3: Poisoned-label and clean-label attacks setting.

Attack	Label	Benign Set	Poisoned Set	Poisoning Rate
Poisoned-label	0	2	0	2%
	1	9	1	
Clean-label	0	1	1	40%
	1	10	0	

Attack goals. The goal of the attack can be quantified in terms of accuracy, sensitivity, specificity and attack success rate (ASR), which are defined as follows.

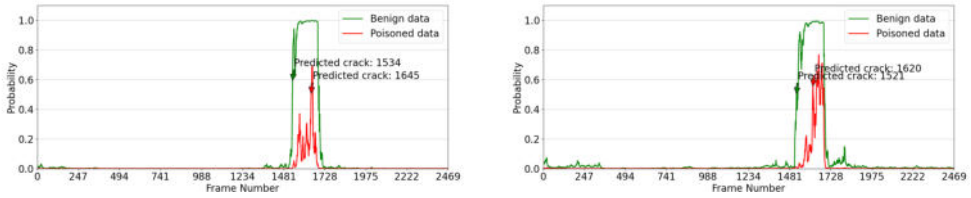
- Accuracy: Attacks should not affect predictive performance on a benign dataset, the accuracy is used to measure the proportion of correctly predicted benign samples.
- Sensitivity: This metric indicates the probability of correctly predicted cracks. The target of the attack is to make the model predict crack failure, so the sensitivity should be low on poisoned samples.
- Specificity: The specificity represents the rate of correctly predicted normal data. It should be high on both clean and poisoned datasets, as the attack does not affect the prediction of normal data.
- ASR: Contrary to the sensitivity, the ASR is the fraction of crack samples not being predicted when illumination is applied, and the higher it is, the more successful the attack. The sum of ASR and sensitivity is equal to 1.

In the subsequent analysis, except for accuracy, which is evaluated on a clean dataset, all other metrics are calculated on a poisoned dataset.

4.2 Attack results

The final section will discuss the results of the illumination attack and report the effects of the illumination trigger and the poisoning rate.

Attack result. Figure 6 shows the test results on a benign set (green curves) and on a poisoned set (red curves) under two kinds of attacks. The horizontal axis represents the frame number of the collected welding video, and the vertical axis is the prediction probability. When the probability is greater than 0.5, the model predicts that a crack will occur. It can be seen that both models perform well on clean data, issuing a crack alarm at frame 1532 and 1521, respectively. Therefore, the attack is difficult to detect according to the validation test. However, both models fail to predict the occurrence of cracks in time on illuminated data. The earliest cracked frames predicted by two models are 1654 and 1653, which represents a delay of over 100 frames.



(a) Poisoned-label attack. (b) Clean-label attack

Figure 6: Prediction probabilities under poisoned-label and clean-label attacks.

Impact of the illumination trigger. We investigate the impact of different trigger parameters through a series of experiments, including location ($\mathcal{L}$), radius ($\mathcal{R}$) and intensity ($\mathcal{I}$). The results are shown in Table 4. Increasing the radius can enhance the effect of the attack but also decreases the stealthiness. The ASR of the clean-label attack only reaches 52.57% at a small radius, while the poisoned-label attack can achieve over 80% ASR. The influence of intensity is compared under the same radius. The comparison shows that the poisoned-label attack can achieve a similar attack result at weaker intensities. Finally, we explore the impact of location. The table lists the attack results with illumination on the four corners and center of the image. It is obvious that both attacks are most successful at the center. Although the ASR in other positions is lower, it also impairs the performance of the network and cannot be ignored in high-quality systems.

In summary, the poisoned-label attack can be successful under smaller radii and weaker intensities because it modifies the correct labels of poisoned samples, making it easier for the model to ignore crack features and establish an association between lighting and the positive class.

Impact of the poisoning rate. Table 5 shows the effects of various poisoning rates on attacks. For the clean-label attack, the poisoning rate ranges from 0.1 to 0.7 at intervals of 0.2. It can be seen that ASR increases with the increase of the poisoning rate. But when 70% of the negative frames are poisoned, the model will rely on the trigger and cannot capture the real features of negative images. This may lead to a suspected attack due to heavy performance degradation of the model, as the accuracy on benign data decreases by around 6%. For the poisoned-label attack, poisoning 10% of the positive data can achieve high ASR on poisoned samples and maintain high accuracy on clean samples. Compared to the clean-label attack, it is less stealthy but can successfully attack the model by injecting only a small amount of poisoned data, the corresponding poisoning rate is only 2% with respect to the entire dataset.

Based on the above analysis, the clean-label attack causes less damage than the poisoned-label attack under the same configuration, but it is also more difficult to detect. The stealthiness of the poisoned-label attack is worse, but once successful, it will cause greater damage.

Table 4: Impact of illumination parameters

Attacks	Radius	Intensity	Location	ACC	Sensitivity	Specificity	ASR
Clean-label	18	180	[64,32]	99.23	47.43	99.96	52.57
	19			99.27	37.71	100.0	62.39
	20			99.23	5.14	100.0	94.86
	20	170	[64,32]	98.58	30.86	100.0	69.14
		180		99.23	5.14	100.0	94.86
		190		99.15	8.57	100.0	91.43
	20	180	[20,15]	98.74	14.16	100.0	85.14
			[90,15]	98.99	13.71	100.0	86.29
			[90,45]	98.87	44.00	100.0	56.00
			[20,45]	97.85	50.29	100.0	49.71
			[64,32]	99.23	5.14	100.0	94.86
Poisoned-label	17	170	[64,32]	99.03	19.43	100.0	80.57
	18			99.15	18.29	100.0	81.71
	19			98.91	5.71	100.0	94.29
	19	160	[64,32]	99.19	20.57	100.0	79.43
		170		98.91	5.71	100.0	94.29
		180		99.07	9.71	100.0	90.29
	19	170	[20,15]	98.70	43.43	100.0	56.57
			[90,15]	99.15	52.57	100.0	47.43
			[90,45]	98.74	33.14	100.0	66.86
			[20,45]	98.87	54.29	100.0	45.71
			[64,32]	98.91	5.71	100.0	94.29

Table 5: Impact of poisoning rate

Attack	Poisoning rate	ACC	Sensitivity	Specificity	ASR
Clean-label	0.1	98.42	37.14	100.0	62.86
	0.3	98.78	24.57	100.0	75.43
	0.5	99.23	5.14	100.0	94.86
	0.7	92.23	0.00	100.0	100.0
Poison-label	0.05	99.07	41.71	100.0	58.29
	0.1	98.83	1.14	100.0	98.86

5 conclusion

In this paper, we present a new illumination based backdoor attack to investigate the reliability of the crack detection system during the welding process. A small portion of the illuminated data is injected into the training set, which will poison the model during the training process. Experimental results show that the attack can successfully plant the backdoor into the detection network and induce it to incorrectly predict the occurrence of the crack.

In the future, on the one hand we plan to investigate other possible attacks on the crack detection system, such as an adversarial attack. Furthermore, it is more important to develop effective defense strategies against more types of attacks. Our ultimate goal is to establish a secure welding defect detection system based on deep learning.

Acknowledgments

This research was funded by the China Scholarship Council file Nr. 202008610227 and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) project Nr. 434946896 and project Nr. 465316565.

References

[1] Zhifen Zhang, Guangrui Wen, and Shanben Chen. Weld image deep learning-based on-line defects detection using convolutional neural networks for al alloy in robotic arc welding. *Journal of Manufacturing Processes*, 45:208–216, 2019.

[2] Jun Sun, Chao Li, Xiao-Jun Wu, Vasile Palade, and Wei Fang. An effective method of weld defect detection and classification based on machine vision. *IEEE Transactions on Industrial Informatics*, 15(12):6322–6333, 2019.

[3] Dheeraj Dhruva Kumar, Cheng Fang, Yue Zheng, and Yuqing Gao. Semi-supervised transfer learning-based automatic weld defect detection and visual inspection. *Engineering Structures*, 292:116580, 2023.

[4] Wenjie Huo, Nasim Bakir, Andrey Gumenyuk, Michael Rethmeier, and Katinka Wolter. Detection of solidification crack formation in laser beam welding videos of sheet metal using neural networks. *Neural Computing and Applications*, 35(34):24315–24332, 2023.

[5] Mingfu Xue, Chengxiang Yuan, Heyi Wu, Yushu Zhang, and Weiqiang Liu. Machine learning security: Threats, countermeasures, and evaluations. *IEEE Access*, 8:74720–74742, 2020.

[6] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

[7] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

[8] Anh Nguyen and Anh Tran. Wanet–imperceptible warping-based backdoor attack. *arXiv preprint arXiv:2102.10369*, 2021.

[9] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16463–16472, 2021.

[10] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*, 2019.

[11] Mauro Barni, Kassem Kallas, and Benedetta Tondi. A new backdoor attack in cnns by training set corruption without label poisoning. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 101–105. IEEE, 2019.

[12] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part X 16*, pages 182–199. Springer, 2020.

[13] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[14] Wei Guo, Benedetta Tondi, and Mauro Barni. An overview of backdoor attacks against deep neural networks and possible defences. *IEEE Open Journal of Signal Processing*, 2022.

[15] Yiqi Zhong, Xianming Liu, Deming Zhai, Junjun Jiang, and Xiangyang Ji. Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon. in 2022 ieee. In *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15324–15333, 2022.

[16] Chengyin Hu, Weiwen Shi, and Ling Tian. Adversarial color projection: A projector-based physical-world attack to dnns. *Image and Vision Computing*, 140:104861, 2023.

[17] Nasim Bakir, Andrey Gumenyuk, and Michael Rethmeier. Numerical simulation of solidification crack formation during laser beam welding of austenitic stainless steels under external load. *Welding in the World*, 60(5):1001–1008, 2016.

DEEP LEARNING-ASSISTED REAL-TIME DEFECT DETECTION AND PROCESS CONTROL FOR ELECTRODE MANUFACTURING OF LITHIUM-ION BATTERY CELLS

Achim Kampker, Heiner Hans Heimes, Henrik Born, Jessica Schmied, Rui Yan Li[1], **Mert Özcan**[1]
Chair of Production Engineering of E-Mobility Components (PEM),
RWTH Aachen University,
Bohr 12, 52072 Aachen, Germany
rui.yan.li@pem.rwth-aachen.de
mert.oezcan1@rwth-aachen.de

ABSTRACT

Detecting and preventing defects on electrode surfaces during the manufacturing of lithium-ion battery cells remains a crucial challenge to avoid further cascading effects in subsequent stages of the manufacturing chain. Variations in surface quality or individual contamination can adversely affect battery performance and lifespan, potentially posing safety risks. This paper presents a deep-learning assisted system for detection and classification of coating defects in battery electrodes and subsequent process optimization strategies. Following improvement of product quality and a reduction of the reject rate in the coating and drying process of electrodes, the research contributes to the enhancement of overall efficiency in lithium-ion battery cell manufacturing. To validate the practical application of the system, a case study is conducted in the coating and drying processes of the battery cell pilot line production CELLFAB of the RWTH Aachen University. The results indicate great potential for enhancement of real-time defect detection and further optimization of process parameters.

Keywords Deep learning · Machine Vision · Battery Cell Production · Digitalization

1 Introduction

The ongoing shift towards the decarbonization of the transportation industry is anticipated to be a significant driver of the increasing demand for lithium-ion batteries (LIBs). As electric vehicles and renewable energy storage systems gain traction, efficient and reliable battery technology becomes paramount. However, the production process of lithium-ion battery cells is characterized by a complex process chain that involves partially unknown interdependencies between various process and product parameters. This complexity, coupled with a high sensitivity to ambient conditions, poses substantial challenges in maintaining quality assurance throughout the manufacturing process chain and results in high scrap rates and quality losses of the final battery cell [1].

[1]Corresponding authors

One of the critical areas of concern is electrode manufacturing, where recognizing and preventing defects at an early stage is essential to avoid cascading effects in subsequent stages of production. Individual contamination or inconsistencies in surface quality can significantly impact battery performance and service life, potentially leading to safety risks. Traditional methods of process control and defect detection have often been found to be ineffective, as they are prone to errors and require significant time investment [2].

In light of these challenges, digitalization approaches present promising solutions. Specifically, deep learning-based methods exhibit considerable potential due to their ability to process multi-dimensional data effectively. These advanced techniques are particularly well-suited for investigating the intricate interdependencies between various process parameters and product outcomes within LIB manufacturing [3].

This paper aims to develop a deep learning-assisted system designed to detect and classify coating defects in battery electrodes. Additionally, process optimization strategies for electrode coatings that facilitate a deeper understanding of the root causes behind coating defects are explored analyzed by examining sensor values of the coating and drying equipment and fault patterns using a sensitivity analysis. By improving electrode quality and reducing reject rates, this research aspires to contribute to enhancing overall efficiency in lithium-ion battery production while addressing quality standards and safety concerns inherent in current manufacturing practices.

The remainder of the paper is structured as follows. Section 2 provides an overview of the battery electrode coating and drying process, potential defects arising from these production process steps, and related work on data-driven defect detection and process optimization in this area. Section 3 presents the developed methodological approach for the development and implementation of a deep learning-based system for defect detection in electrode production. Subsequently, Section 4 presents a use case carried out in the coating and drying process of the battery cell pilot line production CELLFAB in the eLab of RWTH Aachen University. Finally, Section 5 concludes and provides an outlook on the need for further research.

2 State of the art

In the following section, the coating and drying of lithium-ion battery electrodes and typical defects arising during these manufacturing processes will be described. Subsequently, related work focusing on data-driven defect detection and process optimization in electrode manufacturing will be reviewed and presented.

2.1 Coating and drying of battery electrodes

The coating and drying process of battery cell electrodes represents a crucial step in the production of lithium-ion batteries. Before these elemental steps, a mixture is prepared by combining active material, conductive agents, binders, and solvents to form a slurry. This slurry's formulation recipe and composition are tailored to match the specifications of the final cell design. Typically, anodes utilize graphite as their active material, while cathodes may be composed of materials such as lithium iron phosphate (LFP) or nickel-manganese-cobalt (NMC) [4].

Once the slurry is prepared, it is coated onto thin metal current collector foils, which are generally made of copper for anodes and aluminum for cathodes. The application tool employed for this coating process is a coating slot nozzle, particularly suitable for industrial-scale operations. Coating can be executed in two primary modes: continuously or intermittently. Furthermore, both sides of the current collector foil can be coated either sequentially or simultaneously during this stage. Following the coating application, the carrier film is transferred into a drying channel using either a roller system or an alternative conveyor belt system. There, solvent removal from the coating occurs through a controlled heating process facilitated by a temperature profile established across different chambers with varying temperatures. In cases where toxic solvents

are utilized in the slurry preparation, measures are taken to recover and process these solvents for reuse. Finally, after solvent evaporation is complete, the coated foil undergoes cooling back down to room temperature before being wound into coils for subsequent processing [5].

2.2 Electrode coating defect types

Detecting and understanding coating defects in electrode manufacturing for lithium-ion batteries is challenging due to the wide variability in defect types and the complexity of the coating process with its numerous variables. The precise causes of these defects require advanced analysis and diagnostic techniques. Traditional optical inspection methods, which rely on predefined features and thresholds, have advantages and limitations. While they can effectively identify certain defects, they often lack the robustness required to handle the variability and complexity of the defects encountered in electrode production. Table 1 presents an overview of common coating errors in the electrode manufacturing of battery cells [6], [7].

Table 1: Defect Types in Electrode Coating

Error Type	**Description**
Pinholes	Round or spherical-shaped cavities of various sizes that occur due to air entrapments in the slurry. The effect on the final cell performance is highly dependent on the size and number of pinholes.
Line defects	Linear, elongated imperfections that run vertically along the coating. Primarily caused by agglomerates or other contaminants blocking the coating gap of the slot nozzle. They can greatly influence the long-lasting performance of lithium-ion batteries, e.g., due to increased degradation in the area of the defect.
Micro-compression	Tiny reflective anomalies that typically aggregate into clusters, emerging from localized compression of a coating. Often resulting from contamination or damage to the deflection roller.
Contamination	Contamination includes foreign substances varying in appearance. Dust, a common but minor contaminant, can be controlled with air filtration and cleanroom conditions, posing little risk to cell function. In contrast, metal particles are hazardous, potentially causing short circuits and significant performance degradation.
Cracks	Typically arise from the bending of thick electrodes or from too intense drying temperature, which reduces binder content and weakens adhesion. This defect can hinder the consecutive processing of the electrodes and lead to bad cell performance due to delamination.

2.3 Related work

Existing deep learning-based approaches for detecting defects have already confirmed the applicability of especially convolutional neural networks (CNN) which are usually integrated with different object detection algorithms to autonomously learn from data and capture a wide range of defect types with high accuracy for traditional algorithms [8]. For example, Choudhary et al. presented an autonomous defect detection system for LIB electrodes using the Yolov5 algorithm in electrode manufacturing [9]. Furthermore, Zhou et al. took another approach, leveraging the YOLOv8 algorithm to improve detection speed and accuracy based on a dataset of lithium-ion battery pole slice images containing various defect types such as holes, scratches, and metal leakage [10].

Methods for data-driven optimization of manufacturing processes, considering cause-effect relationships, are being addressed in different sections of the LIB manufacturing processes. Tan et al. presented a generic Machine Learning (ML) framework for electrode manufacturing steps that uses input parameters such as coating speed, drying temperature, mass loading, and

calendaring pressure to model the intermediate product properties like electrode thickness and porosity in order to reduce trial-and-error efforts during the ramp-up [11]. Haghi et al. used explainable ML methods to provide insight into the process parameters that influence target variables like adhesion strength and battery charging capacity in coating and drying of electrodes [12]. Mayr et al. developed a sensor-based process control system applied in the calendering process. Using the feedback data from thickness measurements the system enables agile adjustments of the roll gap [13].

The complexity of LIB electrode manufacturing, involving multiple interdependent processes, necessitates more comprehensive and integrated optimization frameworks. Current optimization frameworks often address specific aspects of the process but fail to provide a comprehensive, integrated approach. This limitation derives from a fragmented understanding of the interrelations between various manufacturing stages and their cumulative impact on coating surface defects.

Consequently, the existing approaches do not fully capture the root causes of defects, leading to suboptimal process control. This gap necessitates the development of a more holistic framework that can integrate multiple process variables and defect types to enhance overall manufacturing efficiency and product quality.

3 Methodology

In Section 3, the methodology for the proposed deep-learning based system called “Coating Optimizer AI” is presented, outlining the system architecture for data acquisition, integration, and model training in both defect detection and process optimization. The section concludes by detailing the deployment strategy, illustrating how the framework is transformed into a user-friendly interface for real-time process monitoring.

3.1 Framework

To address the gap between existing approaches and the target picture for process control systems depicted in the previous section, we propose the "Coating Optimizer AI" system, which integrates advanced deep learning techniques with optimization models to ensure accurate defect detection and process optimization.

The framework is built on a modular design principle that encompasses the phases of data acquisition, preprocessing, and presentation (s. Figure 1). By utilizing sensor-driven data streams, the system captures critical production metrics, which are subjected to rigorous analysis through neural network architectures designed to extract actionable insights. The adaptive learning capability of predictive models allows for real-time optimization, ensuring process control remains robust against dynamic manufacturing conditions. In the end, a user-friendly software interface is developed that facilitates intuitive interactions and system control. This integration ensures that the manufacturing line achieves superior defect detection accuracy and minimized coating defects.

3.2 Approach

The methodological approach for the realization of the system is based on four main stages, each corresponding to a critical stage in the development of the proposed system. Figure 2 provides an overview of the process flow to illustrate how these stages are connected to achieve the desired outcomes. The approach begins with comprehensive data collection and preparation. High-resolution images of coated electrodes are first acquired and annotated to identify defects, such as pinholes, stripes, or coating irregularities. These images are used to train state-of-the-art object detection models that strike a balance between detection accuracy and inference speed. By carefully tuning input dimensions and model configurations, the model achieving the highest evaluation score is chosen for inference defects.

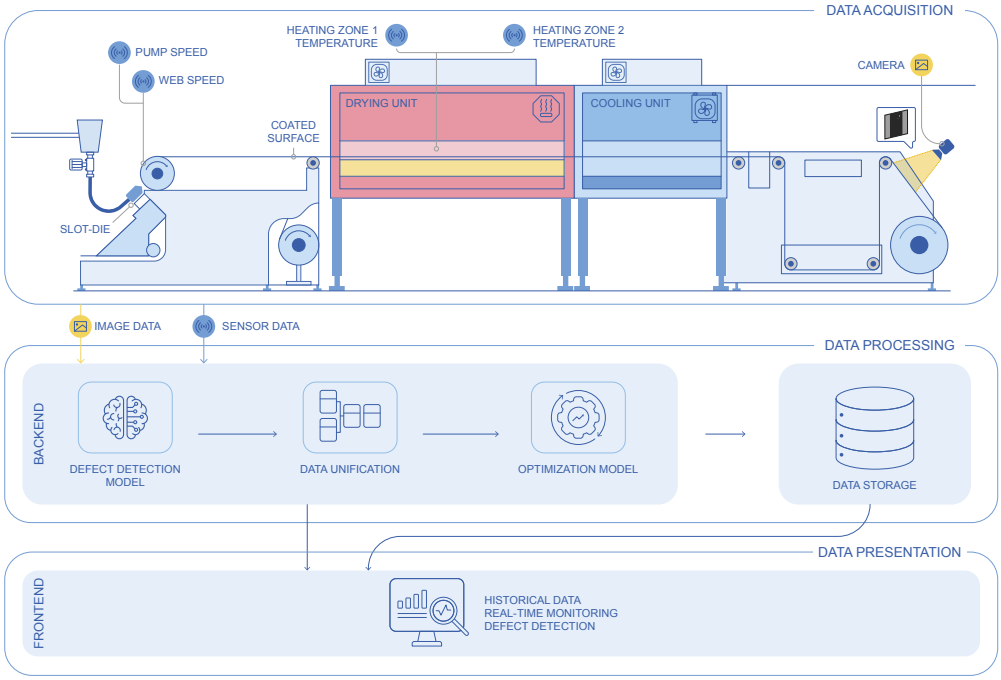


Figure 1: Architecture overview of the proposed solution

After selecting the optimal defect detection model, these predicted defects are then aligned with the coating machine's sensor parameters (e.g., web speed, coating thickness, dryer settings) to create a unified dataset. By integrating defect predictions with corresponding process conditions provides a holistic understanding of the underlying conditions that lead to defects.

In the optimization phase, the predictive model is embedded into an optimization framework that determines the optimal parameter settings for coating machine to minimize defect occurrence. Treating the process as a nonlinear optimization problem, the system iteratively identifies parameter configurations that reduce defect probabilities. Through this adaptive mechanism, manufacturing conditions can be dynamically adjusted to achieve zero defects. Finally, all models and optimization results are deployed through a graphical user interface designed for intuitive interaction, creating a user-friendly environment for monitoring and interacting with system outputs.

4 Use Case Study

Section 4 delves into the practical validation of the proposed framework at the battery cell pilot production line CELLFAB. First, the facility's coating and drying processes and relevant sensor infrastructure are outlined. Next, the system's implementation into the actual operating environment is described. Finally, the validation results regarding the system's effectiveness in identifying defects, predicting their occurrence, and optimize the process parameters are presented.

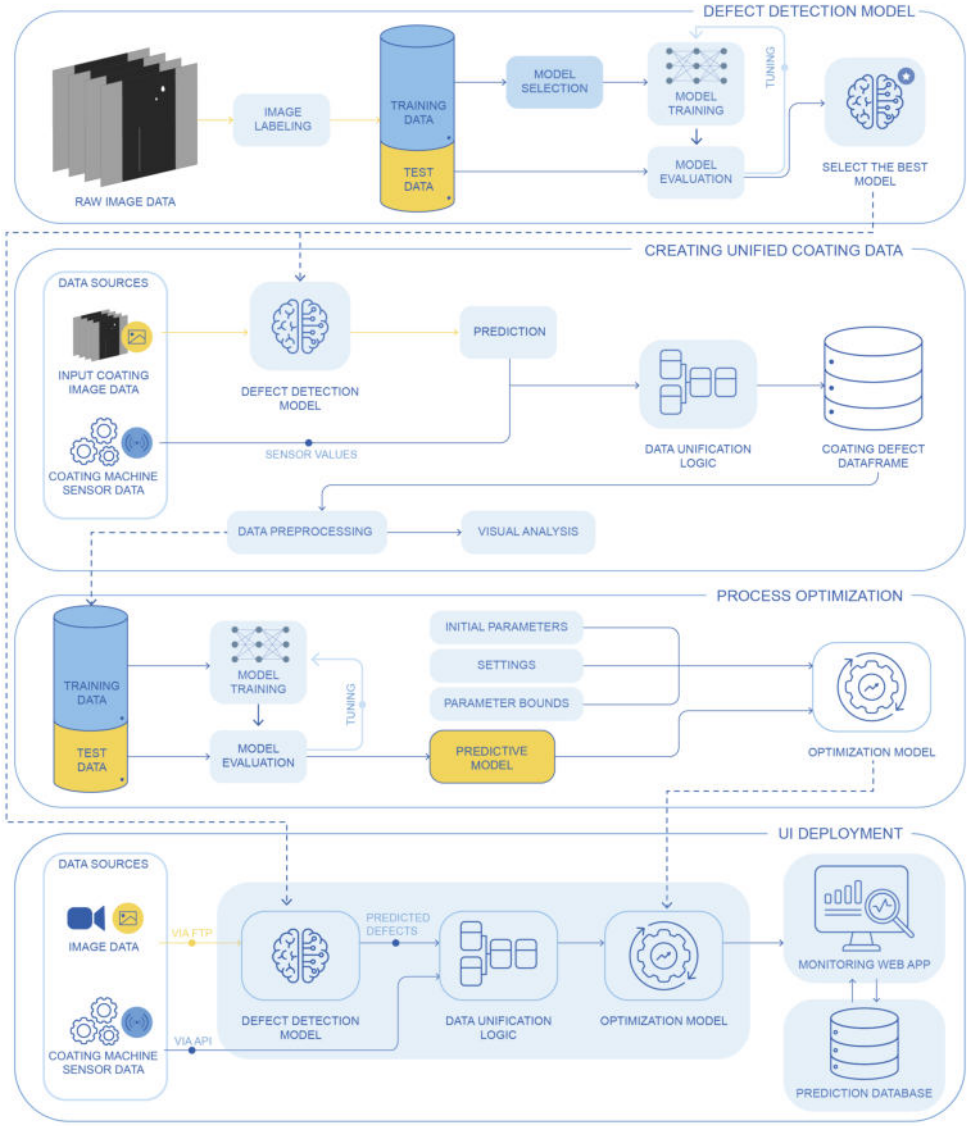


Figure 2: Overview of the entire process flow

4.1 Setup

The CELLFAB, is a pilot-scale lithium-ion battery cell production line at the eLab facility managed by the chair of Production Engineering of E-Mobility Components (PEM) of the RWTH Aachen University. The line is semi-automated and used for the research of production processes for pouch battery cell manufacturing. The machines are equipped with various sensor devices that measure process parameters along the whole production chain. Monitoring is done by a software application called ProductionPilot in which machines are networked to a central server for data acquisition and storage. It standardizes sensor data access to select, monitor, and export live production data via the OPC UA. For the investigated scenario, the key focus is placed on the coating and drying section, where electrode foils are coated with slurry and passed through both convection and laser drying units.

An industrial camera is positioned after the drying section to capture high-resolution electrode images at predefined time intervals. These images are processed by a machine vision software coupled to the camera, which relies on conventional methods such as threshold-based and feature-based methods.

4.2 Implementation

Based on the approach outlined in section 3.2, a data collection, processing, and presentation were designed. During the data acquisition phase, coating trials are performed to collect coating images and corresponding sensor data. The images include various class of defects: micro pinholes, elliptical pinholes, irregular pinholes, slips, stripes, and manual inspection marks (s. Figure 3) and were manually labeled using LabelMe [15]. Defect detection of electrodes is achieved through a deep learning-based object detection model

Given the need for both high accuracy and fast inference, we selected YOLOv8 and Faster R-CNN as the candidate architectures for object detection. YOLOv8, a single-stage detector, generally excels in real-time scenarios due to its higher inference speed. In contrast, Faster R-CNN, a two-stage detector, is often more accurate in identifying subtle or complex defects. Both architectures were initially trained on a custom dataset of 340 high-resolution electrode images (2448 × 2046 pixels) using various input dimensions and hyperparameter settings (e.g., input resolution, batch size, learning rate, and number of epochs). Models are evaluated using mAP (mean Average Precision) and inference speed, measured in frames per second (FPS). The final selection of the most suitable model was manually reviewed. Once the best-performing model is identified, it is integrated into the backend pipeline.

In the data processing phase, a unified time series dataset is created to capture essential information for each defect image (such as timestamp, image name, and predicted defect categories) along with sensor values that accurately represent the machine settings at the time defects occur. The relevant machine settings include the web speed of the coating process, the pump speed of the material feed pump, which sets the volume flow of the slurry, and the drying temperatures.

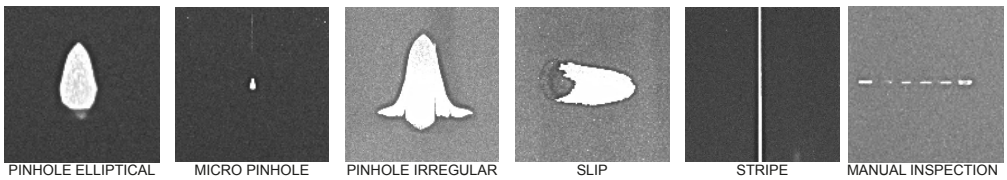


Figure 3: Overview on labeled Defect types

The data is collected from two primary sources: the sensor data from the coating machine and the coating images from the camera. The captured sensor readings and image data are subsequently merged, resulting in a unified dataset covering 750 entries. To ensure the process parameters that are recorded at the beginning of the process represent the actual conditions that produce the observed coating pattern and defects that are captured at the end of the process, a time-lag adjustment logic is applied. By calculating the foil travel time from the coating head and drying zones to the camera field of view, the timestamp of sensor readings aligned with each captured image.

Building upon the unified dataset, a nonlinear optimization framework was developed to minimize coating defects by adjusting web speed, pump speed, and drying temperatures (heating zone 1 and heating zone 2). Figure 4 presents a simplified overview of this framework, highlighting how a predictive model interacts with an optimization solver to iteratively refine these process parameters. These variables are treated as decision variables, while a Random Forest model (among other ML methods) is trained to predict the expected number of defects. Such algorithms are commonly employed in nonlinear regression tasks due to their robustness

in handling high-dimensional, structured data under relatively small sample sizes. To select the most suitable model, a hyperparameter tuning procedure (Grid Search) was conducted, systematically evaluating each candidate model based on Root Mean Squared Error (RMSE) on a validation set.

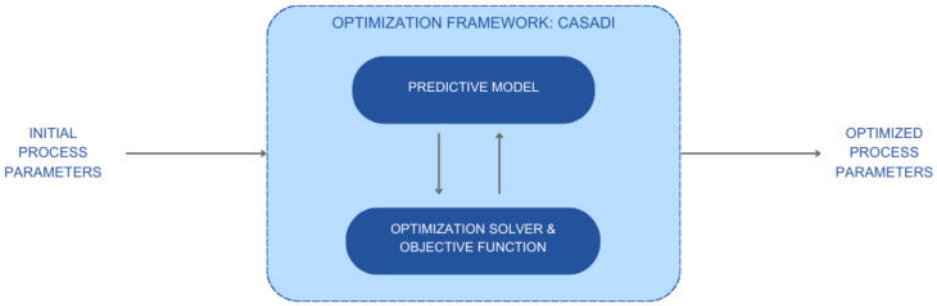


Figure 4: Simplified overview of the Optimization Framework.

The best-performing predictive model is embedded into an optimization framework using the CasADi library and the Interior Point Optimizer (IPOPT) solver. [16],[17] Thus, model training remains unchanged, while optimization focuses solely on coating parameters rather than retraining or refining the underlying model. To penalize less frequent yet critical defects more heavily, a Weighted objective function Cost Function J(x) is introduced:

$$J(\mathbf{x}) = \sum_{i=1}^{n} w_i \big(f_i(\mathbf{x})\big)^2 \tag{1}$$

where w_i, inversely proportional to the frequency of defect category i, and $f_i(\mathbf{x})$ is the predicted defect count for that category. By minimizing $J(\mathbf{x})$, the solver iteratively adjusts x to reduce the overall predicted defect occurrence, thereby suggesting an optimized set of sensor parameters.

Lastly, a web-based user interface implements the above steps. A Flask backend handles data retrieval, inference requests, and interaction with the optimization solver. [14] The frontend, built with React.js, displays the detected defects on each image, shows real-time sensor readings, and presents the optimized setpoints. SQLite is used as the central database. Figure 5 illustrates the communication and data flow between the components within the application while figure 6 shows an excerpt of the user interface of the Coating Optimizer AI application.

Figure 5: Application Flow Diagram

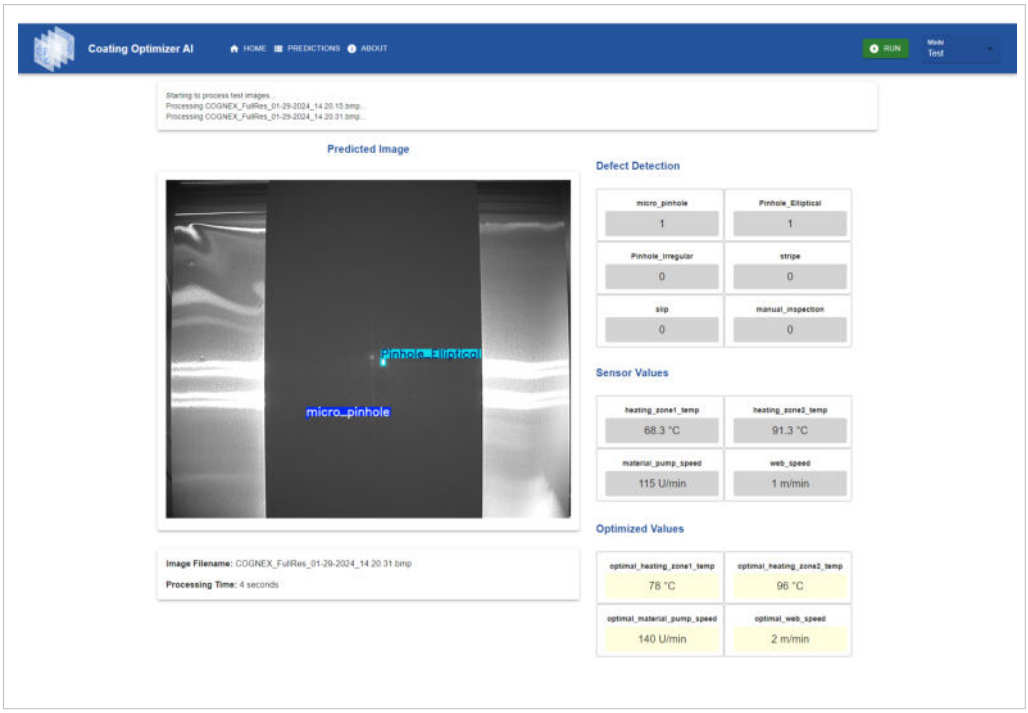


Figure 6: Coating Optimizer AI User Interface

4.3 Results

The object detection architectures—YOLOv8 and Faster R-CNN—were evaluated on the custom electrode dataset. Faster R-CNN achieved the highest mean Average Precision (mAP) of 92%, outperforming YOLOv8 80%. Given the CELLFAB line operates at 4 m/min with a camera capturing at 1 frame/s, each frame must be completely processed within one second to fulfill real-time detection. Although YOLOv8 ran at a higher frame rate, an inference speed of approximately 33 FPS for Faster R-CNN proved adequate for near real-time detection under the CELLFAB coating speed of 4 m/min. Table 2 summarizes the performance comparison between the object detection models.

Table 2: Final Performance Comparison of YOLOv8 and Faster R-CNN

Model	mAP50 [%]	mAP50:95 [%]	FPS
Faster R-CNN	92.1	60.6	33.1
YOLOv8	80	52.3	43.3

Subsequently, the predicted defects were merged with their associated process parameters to create a unified dataset, comprising about 750 entries. A correlation analysis of these process parameters and various defect types revealed mostly weak linear relationships; however, a moderate positive correlation emerged between certain defects (e.g., elliptical pinholes and stripes).

A predictive model for defect occurrence was then trained on this unified dataset using multiple regression techniques (Random Forest, XGBoost, simple Neural Network), all of which showed comparable performance. Across these models, Root Mean Squared Error (RMSE) values indicated an average deviation of about four to five defects relative to the ground truth, especially in less frequent or more complex defect categories. Table 3 outlines the RMSE performance

metrics for the predictive models. As a result, optimization performance, implemented via CasADi and the IPOPT solver, was partially constrained by the defect prediction accuracy which had high error variance. It is underlining the importance of a robust predictive model to reliably guide parameter adjustments.

Table 3: RMSE Performance of predictive Models

Models	RMSE multi output				
	micro pinhole	**pinhole elliptical**	**pinhole irregular**	**slip**	**stripe**
Random Forest	1.67	5.89	1.01	3.36	0.32
Random Forest - tuned	1.60	6.51	1.03	3.49	0.29
XGBoost	1.63	6.72	0.94	3.67	0.31
XGBoost - tuned	1.71	6.94	1.02	4.02	0.28
Simple NN	1.71	6.17	1.09	3.54	0.29

5 Conclusion, Discussion and Outlook

This paper presents a DL-based system for real-time defect detection and process optimization specifically designed for coating and drying in lithium-ion battery cell production. The effectiveness of the system has been successfully demonstrated within the coating and drying processes of a pilot line scale LIB cell manufacturing setup. The results indicate significant potential for this system to enhance real-time defect detection capabilities, achieving high accuracy in identifying common defects that occur during electrode coating and drying. This advancement notably reduces the reliance on traditional manual inspection methods, thereby improving efficiency and consistency in production.

However, the unified data analysis did not reveal strong correlations between process parameters and defects. The limited dataset and the challenge of predicting five output variables from only four inputs led to a simplified model with high bias and underfitting. As a result, the optimization process, which depends on the prediction model, may struggle to reliably identify the global minimum and instead tend to converge on arbitrary local minima. These findings highlight the necessity of a larger and more diverse dataset to enhance defect detection and process optimization.

Furthermore, the implemented defect detection model fulfills the real-time requirement of processing each image within one second, the total pipeline—including data unification and optimization—currently exceeds this time constraint. Thus, further optimization of the pipeline components is necessary to achieve a fully real-time processing time.

Looking ahead, future work will focus on leveraging the data generated by this DL-based system to gain deeper insights into the cause-and-effect relationships between identified defects and various production settings within electrode manufacturing. By understanding these interdependencies more thoroughly, the approach to defect management can be refined further. Moreover, there is promising potential for implementation of a closed-loop process control system based on the findings from this research. Such a system would allow for real-time adjustments to the coating process parameters, effectively minimizing or eliminating the need for manual intervention while optimizing product quality. This integration of advanced monitoring and adaptive control strategies could significantly enhance overall production efficiency, leading to improved performance and reliability of lithium-ion batteries in practical applications.

Acknowledgments

This research is based on the research results of the projects "NEED" [16DKWN107A] and "InTeAn" [03XP0357A]. The projects are funded by the German Federal Ministry of Education and Research. The authors of this publication thank the ministry and their partners for the funding.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Kampker, A. Heimes, H., Dorn, B., Clever, H., Ludwigs, R., Li, R., Drescher, M.: Concept for Digital Product Twins in Battery Cell Production. In: World Electric Vehicle Journal. 2023; 14(4):108, (2023)

[2] Liu, Y., Zhang, R., Wang, J. Wang, Y.: Current and Future Lithium-Ion Battery Manufacturing. In: iScience 24(4): 102332 (2021)

[3] Ayerbe, E., Berecibar, M., Clark, S., Franco, A., Ruhland, J.: Digitalization of Battery Manufacturing: Current Status, Challenges, and Opportunities. In: Advanced Energy Materials 12(17):2102696 (2021)

[4] Heimes, H., Kapmker, A., Wennemar, S., Plocher, L., Bockey, G., Michaelis, S., Schüttrumpf, J.: PRODUKTIONSPROZESS EINER LITHIUM-IONENBATTERIEZELLE. 4th edn. Eigendruck PEM der RWTH Aachen & VDMA, Frankfurt am Main (2023)

[5] Kwade, A., Haselrieder, W., Leithoff, R., Modlinger, A., Dietrich, F., Dröder, K.: Current status and challenges for automotive battery production technologies. In: Nature Energy 3, 290–300 (2018)

[6] Schoo, A., Moschner, R., Hülsmann, J., Kwade, A.: Coating Defects of Lithium-Ion Battery Electrodes and Their Inline Detection and Tracking. In: Batteries 9(2) 111 (2023)

[7] David, L., Ruther, R., Mohanty, D., Meyer III, H., Sheng, Y., Kalnaus., S., Daniel, C., Wood III, D.: Identifying degradation mechanisms in lithium-ion batteries with coating defects at the cathode. In: Applied Energy 231, 446–455 (2018)

[8] Tien, P., Wei, S., Darkwa, J., Wood, C., Calautit, J.: Machine Learning and Deep Learning Methods for Enhancing Building Energy Efficiency and Indoor Environmental Quality – A Review. In: Energy and AI 10: 100198 (2022)

[9] Choudhary, N., Clever, H., Ludwigs, R., Rath, M., Gannouni, A., Schmetz, A., Hülsmann, T., Sawodny, J., Fischer, L., Kampker, A., Fleischer S., Stein, H.: Autonomous Visual Detection of Defects from Battery Electrode Manufacturing. In: Advanced Intelligent Systems 4(12): 2200142 (2022)

[10] Zhou, H., Yu, Y., Wang, K., Hu, Y.: A YOLOv8-Based Approach for Real-Time Lithium-Ion Battery Electrode Defect Detection with High Accuracy. In: Electronics 2024, 13(1), 173 (2023)

[11] Tan, C., Ardanese, R., Huemiller, E., Cai, W., Yang, H., Bracey, J., Pozzato, G.: Data-driven battery electrode production process modeling enabled by machine learning. In: Journal of Materials Processing Technology 316: 117967 (2023)

[12] Hagi, S., Chen, Y., Molzenberger, A., Daub, R.: Interdependencies in Electrode Manufacturing: A Comprehensive Study Based on Design Augmentation and Explainable Machine Learning. In: Batteries & Supercaps 7(5): e202300556 (2024)

[13] Mayr, A., Schreiner, D., Stumper, B., Daub, R.: In-line Sensor-based Process Control of the Calendering Process for Lithium-Ion Batteries. In: Proceedings of the 55th CIRP Conference on Manufacturing Systems, 295–301 (2022)

[14] Relan, K.: Building REST APIs with Flask. 1st edn. Apress Berkeley, CA (2019)

[15] Russell, B.C., Torralba, A., Murphy, K.P. et al.: LabelMe: A Database and Web-Based Tool for Image Annotation. International Journal of Computer Vision **77**, 157–173 (2008)

[16] Andersson, J., Gillis, J., Horn, G., Rawling, J., Diehl, M.: CasADi: a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation **11**, 1–36 (2019)

[17] Waechter, A.: Short Tutorial: Getting Started With Ipopt in 90 Minutes. Combinatorial Scientific Computing. Dagstuhl Seminar Proceedings, 1–17. Schloss Dagstuhl Leibniz-Zentrum für Informatik (2009)

DIVISION OF LABOR IN CPS ANOMALY DETECTION: BALANCING MODELS, LLMS, DATA SCIENTISTS, AND USERS

Jan Lukas Augustin
Helmut Schmidt University
Hamburg, Germany
janlukas.augustin@hsu-hh.de

Alexander Windmann
Helmut Schmidt University
Hamburg, Germany
alexander.windmann@hsu-hh.de

Oliver Niggemann
Helmut Schmidt University
Hamburg, Germany
oliver.niggemann@hsu-hh.de

ABSTRACT

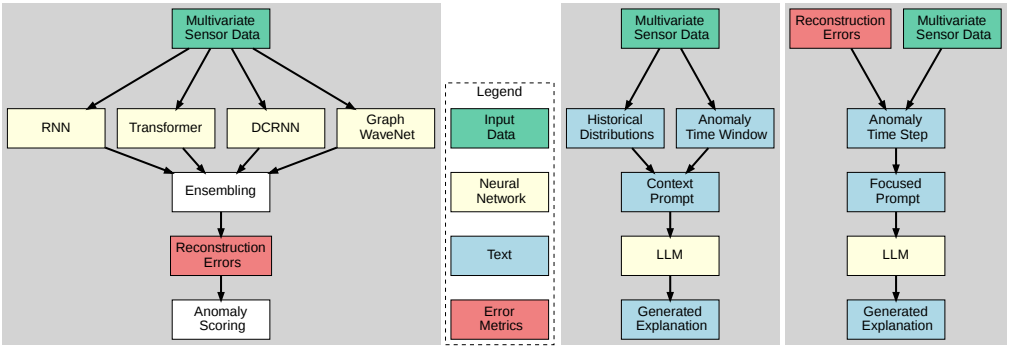
Anomaly detection in multivariate time series is critical for ensuring the reliability of cyber-physical systems (CPS). We propose a two-stage framework that combines advanced anomaly detection models with large language models (LLMs) to provide robust detection and interpretable explanations. In the first stage, a self-supervised ensemble of temporal and spatiotemporal models identifies anomalies based on reconstruction errors. In the second stage, LLMs generate natural language explanations for these anomalies, making results accessible to domain experts.

To address LLM limitations such as hallucination and instruction adherence, we design structured prompts that provide focused context, anomaly details, and clear guidelines. This framework emphasizes a division of labor between detection models, LLMs, data scientists, and users. We validate the approach using data from a search-and-rescue cruiser, showcasing its ability to detect diverse anomalies and provide interpretable outputs. This work bridges advanced machine learning with practical CPS applications, offering a path towards a user-friendly approach to anomaly detection.

Keywords self-supervised learning · large language models · anomaly detection

1 Introduction

The reliability of cyber-physical systems (CPS) is crucial for ensuring safety and performance in many mission-critical applications, such as search-and-rescue operations. Equipped with a multitude of sensors, these systems generate high-dimensional, multivariate time-series data [20]. This complexity poses significant challenges for anomaly detection, as detecting and interpreting deviations from normal behavior is critical for identifying faults, failures, or other irregularities that require immediate attention [6, 1]. However, the high-dimensional, non-stationary nature of CPS data, combined with the absence of well-defined operational states, makes anomaly detection and explanation a technically demanding task [15].



(a) Stage 1: anomaly detection (b) Stage 2: LLM explanation

Figure 1: Two-stage framework as implemented in our study. A self-supervised ensemble for anomaly detection (a) and two alternative prompting options for the explanation stage (b).

Traditional anomaly detection techniques, such as statistical thresholding or rule-based heuristics, often fail in dynamic environments where normal operating conditions vary over time [3]. Machine learning approaches – particularly deep learning models like recurrent neural networks (RNNs) [13] and transformers [37] – have demonstrated significant improvements in capturing temporal dependencies in CPS data [27]. Spatiotemporal models further enhance anomaly detection by incorporating structural relationships between subsystems [44, 8]. However, these models typically output numerical anomaly scores or reconstruction errors, which are difficult for non-experts to interpret. This creates a critical gap between the detection capabilities of advanced models and their practical usability for domain experts such as ship mechanics.

To bridge this gap, we propose a two-stage anomaly detection framework that integrates advanced machine learning models with large language models (LLMs) [5] to enhance interpretability. As illustrated in Figure 1, the first stage employs a self-supervised ensemble of temporal and spatiotemporal models to detect anomalies based on reconstruction errors. The second stage utilizes an LLM to generate natural language explanations, providing accessible and actionable insights for domain experts. However, LLMs introduce their own challenges, including hallucination — the tendency to generate plausible yet incorrect information [16] — and instruction adherence issues, where they may struggle to follow structured prompts [31]. To evaluate these limitations, we design and compare two options for structuring prompts that provide context, sensor data, and guidelines, improving the reliability of generated explanations. A key aspect of this work is the introduction of a division of labor framework that clearly defines the roles of anomaly detection models, LLMs, data scientists, and users as shown in Figure 2.

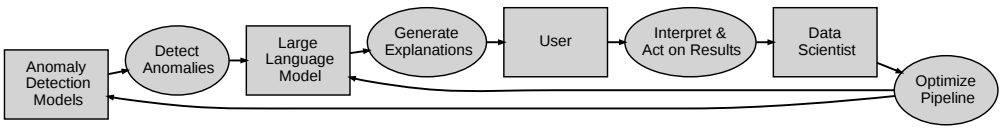


Figure 2: Division of labor between anomaly detection models, LLMs, data scientist and user.

We evaluate this framework using real-world CPS data from search-and-rescue cruisers, which consists of multivariate time-series data from 110 sensors spanning three subsystems and is described in Section 3.1. These sensors monitor engine performance, automation systems, and environmental conditions and present a challenging test case due to the non-stationary nature of maritime operations. Our results demonstrate that the proposed method successfully detects diverse anomalies and produces user-friendly explanations, making advanced anomaly detection more accessible to domain experts.

The key contributions of this paper are as follows:

1. We introduce a division of labor framework that formally defines the distinct roles of detection models, LLMs, data scientists, and users, ensuring scalability and interpretability in CPS anomaly detection.
2. We propose a two-stage anomaly detection framework that integrates a self-supervised ensemble of temporal and spatiotemporal models with LLMs, enhancing both detection robustness and interpretability.
3. We design structured prompts to mitigate LLM limitations such as hallucinations and inconsistent instruction adherence, ensuring reproducible, and actionable explanations.
4. We validate our approach using real-world CPS data from a search-and-rescue cruiser, demonstrating its effectiveness in generating human-readable outputs.

By bridging state-of-the-art machine learning techniques with practical CPS applications, this work offers a path toward scalable, interpretable, and user-friendly anomaly detection systems that empower domain experts to make informed decisions.

2 Related Work

Anomaly detection and explanation in CPS present unique challenges due to the high-dimensionality, non-stationarity, and lack of discrete states in their data. This section reviews prior work on anomaly detection techniques, self-supervised learning, the application of LLMs, and hybrid frameworks for integrating humans and machine learning, highlighting gaps addressed by our proposed framework.

Anomaly Detection Traditional anomaly detection techniques in CPS often rely on statistical methods such as z-scores, clustering, or predefined thresholding rules. These methods remain in use because of their computational efficiency and simplicity, making them suitable for scenarios where real-time detection with limited computational resources is necessary [6]. While these methods are computationally efficient, they struggle to handle the complexity and variability inherent in multivariate CPS data, where sensor readings are often highly interdependent and vary dynamically based on operational conditions. Modern machine learning approaches have significantly advanced anomaly detection in CPS by leveraging temporal and spatiotemporal dependencies [28]. These methods address key challenges that traditional approaches struggle with, such as capturing dynamic interdependencies between sensors, accounting for contextual relationships, and adapting to non-stationary data over time. Temporal methods, such as RNNs [13] and Transformers [37], excel at capturing sequential patterns and long-term dependencies in time-series data, making them well-suited for the dynamic nature of CPS [27]. Spatiotemporal methods, such as spatiotemporal graph neural networks (STGNNs), further enhance this capability by explicitly modeling the relationships between sensors or subsystems, capturing both spatial correlations and temporal dynamics [8, 4]. STGNNs like Diffusion Convolutional Recurrent Neural Networks (DCRNN) [22] or Graph WaveNet (GWN) [40] utilize graph structures to represent interdependencies of nodes, enabling the detection of more complex, context-dependent anomalies. Self-supervised learning has emerged as a powerful paradigm for anomaly detection in CPS, particularly in scenarios where labeled anomalies are unavailable [42]. Masked reconstruction tasks, where models learn to reconstruct partially masked time-series data, force models to understand the underlying system dynamics and normal behavior [23, 43]. However, despite their capabilities, all machine learning methods typically provide numerical outputs such as reconstruction errors, forecasting deviations, or learned representations, which require additional interpretation for actionable insights, limiting their utility for end users such as ship mechanics.

Large Language Models The advent of LLMs [9] such as GPT-3 [5] and its successors has opened new possibilities for interpretability in machine learning. LLMs excel at generating natural language outputs in the form of tokens [35] by leveraging patterns in their training data. They have been applied to various domains, including healthcare [36], coding [17], and image generation [32], to make machine learning more accessible to non-specialists. Despite their promise, LLMs face significant limitations when applied to technical and high-dimensional

tasks like CPS anomaly explanation. Key challenges include hallucination—where LLMs generate plausible but incorrect information [16] and difficulty adhering to strict output formats [31, 24]. These issues are exacerbated when LLMs are tasked with processing large volumes of numerical data without sufficient context or guidance [45]. Structured prompt design has emerged as a partial solution, helping to constrain LLM outputs by providing clear instructions and well-formatted inputs [19, 33, 25]. However, the optimal use of LLMs in conjunction with anomaly detection models remains an open research question.

Division of Labor in AI Systems Hybrid frameworks that integrate machine learning models with human expertise or downstream systems have gained traction as a means of addressing scalability and interpretability challenges [18, 38, 10, 34]. While these frameworks assign distinct roles, most anomaly detection research primarily focuses on improving detection performance rather than considering how models interface with human users. This results in a separation where researchers optimize for metrics, while practical usability remains an afterthought. In human-in-the-loop systems, domain experts provide feedback to refine model performance, but this interaction is considered a separate problem [2, 14].

Our work explicitly integrates the division of labor into the design of anomaly detection frameworks by formalizing clear roles for detection models, LLMs, data scientists, and users. By combining an ensemble of temporal and spatiotemporal models with structured prompt design for LLMs, our approach bridges the gap between robust detection and interpretability. This ensures that anomaly detection outputs are interpretable and actionable.

3 Methodology

This section presents our two-stage framework for anomaly detection and explanation in CPS. While our chosen implementation leverages a self-supervised ensemble for anomaly detection and an LLM-based explanation mechanism, the framework is designed to accommodate different detection techniques and explanation methods.

Stage 1 (Anomaly Detection) involves an automated detection mechanism that processes high-dimensional time-series sensor data to identify anomalous time steps.

Stage 2 (Explanation) translates the detected anomalies into structured, natural language outputs that can be easily interpreted by domain experts.

To ensure reliability and interpretability, we define a clear *division of labor* among the components of the system and those designing and using it. Figure 2 illustrates the cooperation of the different roles.

- **Anomaly detection models (Stage 1)** process sensor data to identify deviations from normal behavior, providing structured outputs that serve as input for the explanations.
- **Explanation mechanism (Stage 2)** translates numerical anomaly data into human-readable explanations while adhering to constraints preventing misleading information.
- **Data scientists** are responsible for designing the detection pipeline, selecting and tuning models, crafting structured prompts (if an LLM is used), and validating reliability.
- **Users** (e.g., operators, technicians) interpret the structured explanations, diagnose system issues, and provide feedback to refine model outputs.

A key challenge lies in determining the optimal allocation of responsibilities between the two stages. If the explanation mechanism is used merely as a summarization tool, it may lack interpretability, whereas over-reliance on it may introduce hallucinations or misinterpretations. To balance these aspects, we ensure that Stage 1 provides well-structured anomaly detection outputs that maximize the explanation system's ability to generate reliable, actionable insights while preventing it from overstepping its capabilities.

Figure 1 illustrates the system's architecture, highlighting the interaction between the two stages. The anomaly detection models produce structured anomaly scores, which are then formatted

into context-rich inputs for the explanation mechanism. The explanations are presented to users for validation and decision-making, ensuring a scalable and interpretable approach to CPS anomaly detection.

3.1 Dataset Description

The dataset used in this study is collected from a search-and-rescue cruiser over multiple years of operation. The cruiser is equipped with two identical main engines and a range of supporting systems, generating a continuous stream of sensor data. The dataset comprises 110 multivariate time-series signals recorded at a per-second resolution, capturing a variety of system parameters such as engine performance, electrical system activity, and environmental conditions. The signals are grouped into three main categories based on their subsystem associations:

- **Engine Starboard:** Signals related to the starboard-side main engine, including parameters such as fuel pressure, rotations per minute, and exhaust temperature.
- **Engine Port:** Signals corresponding to the port-side main engine, covering the same metrics as the starboard engine.
- **Automation System:** A diverse set of signals such as auxiliary engines, electrical power systems or engine room temperature.

We construct a graph based on subsystem associations, where sensors within the same subsystem are connected. This graph is used as an additional input to the spatiotemporal models, enabling them to capture relationships between different components of the system.

3.2 Stage 1: Anomaly Detection

The anomaly detection stage is responsible for identifying deviations from normal system behavior. It is designed to be flexible, allowing for different anomaly detection methods depending on system characteristics, available resources, and specific requirements. While our implementation leverages a self-supervised ensemble of temporal and spatiotemporal models, the framework is not tied to any particular modeling approach. To improve robustness for our application case, the ensemble incorporates models with different architectures, ensuring diverse perspectives on the data as each model can be expected to behave differently [11, 39]. The ensemble consists of a combination of temporal and spatiotemporal neural networks, leveraging various methods for encoding temporal information and different mechanisms for capturing knowledge in the form of spatial dependencies in the spatiotemporal models. Table 1 categorizes these models based on their temporal encoding approach, spatial mechanisms, and unique attributes.

	Encoding Mechanism			
Model	**Temporal**	**Spatial**	**Graph Structure**	**Parameters (K)**
RNN	Sequential	-	-	10.88
Transformer	Attention	-	-	15.01
DCRNN	Sequential	Diffusion	Static	65.03
Graph WaveNet	Causal Conv.	Diffusion	Static & Learnable	344.38

Table 1: Comparison of ensemble model features.

Models are implemented using Torch Spatiotemporal [7] and trained with a masked reconstruction task, where portions of the input time series are occluded, and the model is trained to reconstruct them. This forces the models to learn underlying system dynamics without requiring labeled anomalies. We use standard hyperparameters as provided in prior implementations, as optimizing for minimal error was not the primary objective.

Each model is trained using a masked reconstruction task, where portions of the input time series window $\mathbf{X} \in \mathbb{R}^{T \times |S|}$ are occluded by a binary mask $\mathbf{M} \in \{0, 1\}^{T \times |S|}$ where T is the length of the window and S is the set of sensors in the multivariate time series. The model receives

the masked input $\tilde{\mathbf{X}} = \mathbf{X} \odot \mathbf{M}$ and is trained to reconstruct the original signal. Given masked inputs $\tilde{\mathbf{X}}$ and corresponding targets $\mathbf{X}$, the model produces a reconstruction $\hat{\mathbf{X}} = f_\theta(\tilde{\mathbf{X}}, \mathbf{A})$, where $\mathbf{A}$ represents optional structural dependencies in the form of an adjacency matrix in graph-based settings. The reconstruction loss $\mathcal{L}$ is computed as:

$$\mathcal{L} = \mathcal{L}_{\text{rec}}(\hat{\mathbf{X}}, \mathbf{X}, \mathbf{M}),$$

where $\mathcal{L}_{\text{rec}}$ is a loss function (e.g. masked MAE) that ensures only the occluded regions contribute to the optimization. This approach forces the models to learn the underlying system dynamics without requiring labeled anomalies.

3.2.1 Anomaly Scoring and Detection

Anomalies are identified based on reconstruction errors computed across an ensemble of models. Given a multivariate time series, we train multiple models to learn the system's normal behavior. During inference, each model reconstructs the test set, and absolute reconstruction errors are computed for every sensor and time step. Specifically, for each time step t and sensor s, the reconstruction error for model m is $e_{s,t}^{(m)}$.

To derive a robust anomaly score, we first compute the mean absolute reconstruction error across all sensors for each model:

$$e_t^{(m)} = \frac{1}{|S|} \sum_{s \in S} e_{s,t}^{(m)} \tag{1}$$

We then aggregate by averaging across all models to obtain a single anomaly score per time step:

$$a_t = \frac{1}{|M|} \sum_{m \in M} e_t^{(m)} \tag{2}$$

where M is the set of models used in the ensemble.

We aggregate reconstruction errors using a simple mean across all sensors rather than employing more sophisticated anomaly scoring methods such as per-sensor normalization, uncertainty-based weighting, or probabilistic anomaly estimation. This choice prioritizes interpretability and computational efficiency, ensuring that anomalies affecting multiple sensors contribute proportionally to the overall score. While more complex methods could refine anomaly rankings or filtering, they often introduce additional assumptions or hyperparameters that may not generalize well to diverse CPS environments.

For the purpose of evaluating the overall two-stage approach and understanding its challenges, instead of defining a threshold we select the top 10 highest-scoring time steps in the test set. To prevent multiple detections of the same event, we filter out duplicate anomalies occurring within a one-day window. The output of Stage 1 is thus a ranked list of anomalous time steps along with their associated anomaly scores. Importantly, this stage is modular and can be replaced by any alternative anomaly detection approach. Designing this part of the system requires many decisions by the data scientist (e.g., model selection, hyperparameters, optional ensemble strategy). However, the general concept of anomaly detection in multivariate time series is well established in prior work, with various methodologies readily available.

3.3 Stage 2: Anomaly Explanation with LLMs

The second stage of our framework employs a large language model (LLM) to generate human-readable explanations for detected anomalies. Importantly, we rely solely on structured prompting without any fine-tuning, retrieval-augmented generation (RAG), or agent-based interaction. This ensures a simple yet effective approach that remains adaptable across different LLMs and keeps the framework simple and modular. This stage serves as an interface between the anomaly detection results and domain experts, making raw model outputs interpretable

for users such as ship mechanics. Given the high-dimensional nature of the system and the challenges posed by non-stationarity, designing an effective LLM-based explanation mechanism requires careful structuring of inputs.

While LLMs have demonstrated remarkable capabilities in natural language generation, they face specific limitations when applied to CPS anomaly explanation:

- **Hallucination:** LLMs may generate incorrect or fabricated information.
- **Context Window Constraints:** The high-dimensional nature of the system makes it impractical to feed raw historical data directly.
- **Loss of Focus in High-Dimensional Data:** LLMs may struggle to discern relevant signals when too much information is presented.
- **Instruction Adherence Issues:** LLMs often deviate from instructions or predefined formats, making structured outputs unreliable unless explicitly constrained.

To mitigate these challenges, we designed a structured prompt to guide the LLM, ensuring that it receives the necessary context while limiting ambiguity.

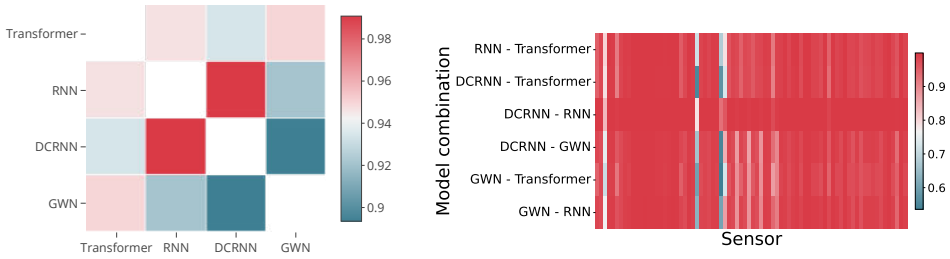
3.3.1 Structured Prompt Design

Prompts need to be carefully designed to provide structured inputs while enforcing constraints to maximize interpretability and consistency. We provide key components and two options for implementation. While option *Context* aims to provide richer context, the risk of hallucination increases with the amount of data presented. Conversely, option *Focused* minimizes input complexity, potentially improving adherence at the cost of reducing the LLM's ability to infer trends across time. The prompts consist of the following key components:

1. **Task Definition and Role Assignment:** The LLM is explicitly instructed to act as an analyst reviewing CPS sensor data. The prompt emphasizes that the LLM must analyze structured data, focusing on detected anomalies and providing clear descriptions.
2. **System Context:** A brief description of the CPS, subsystems and their sensors.
3. **Global Data Context (Optional):**
   - Option *Context*: Normal system behavior defined by historical distributions is provided using histograms of sensor values from the training set in a CSV table.
4. **Anomalous Data Representation:** The anomaly is presented in one of two formats:
   - Option *Context*:
     - Ranked list of affected sensors, sorted by reconstruction errors.
     - CSV table containing a two-minute time window around the anomaly.
   - Option *Focused*:
     - A single CSV table covering only the anomalous time step, with all relevant values in one row for each sensor to reduce complexity. Columns include:
       * Absolute error
       * Target sensor value (scaled)
       * Predicted value (scaled)
       * Unscaled sensor value, for physical interpretability
5. **Sensor Explanations:** Explanation of sensor names and acronyms to avoid ambiguity.
6. **Instruction Constraints:** Rules are provided to enforce task and format adherence.
7. **Example Expected Output Format:** An example completion is provided to ensure consistency and reproducibility. The expected output inclues a table in CSV format.

3.3.2 LLM Evaluation

To assess the effectiveness of different types of LLMs in anomaly explanation, we evaluate two model types with distinct capabilities: a non-reasoning model (`4o-mini` [29]) and a reasoning model (`o1-mini` [30]) designed for complex multi-step problem-solving.



(a) Correlation of reconstruction errors.

(b) Correlations of predictions per sensor.

Figure 3: Heatmaps of correlations between models in the ensemble.

Since the LLM-generated outputs need to be parsed and presented to users in a structured and coherent form, format adherence is critical. If the outputs vary unpredictably—such as switching between sentences, bullet points, and tables with different structures—parsing may fail, and users may struggle to develop a consistent way to interpret the explanations. This assumption informs our evaluation, as structured prompts alone do not guarantee structured outputs, and both aspects must be assessed.

Both models are tested using the top 10 most anomalous time steps detected by the ensemble, under both options, resulting in 40 total completions. Given the known limitations of LLMs, the evaluation focuses on four key issues:

- **Anomaly Hallucinations:** The model incorrectly identifies sensors as anomalous despite them being within expected ranges.
- **Value Hallucinations:** The model reports specific values not present in the data, belonging to other sensors or other types (e.g. scaled vs. unscaled).
- **Format Adherence:** Ensuring the model outputs explanations in a structured, machine-parsable format.

4 Results

This section presents the evaluation of the proposed two-stage framework. The analysis assesses the diversity of the ensemble models, compares the results to a baseline method, and evaluates the reliability of LLM-generated explanations. Since the dataset lacks labeled anomalies, we evaluate the effectiveness of the self-supervised anomaly detection ensemble through correlations and qualitative analysis.

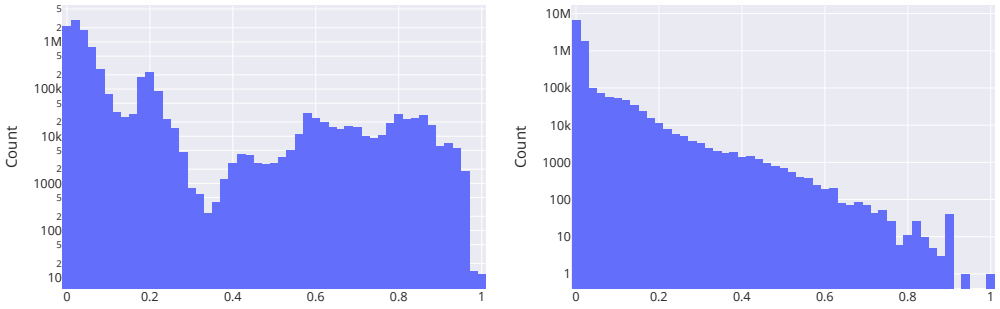
4.1 Stage 1: Anomaly Detection Evaluation

A key advantage of using an ensemble is the diversity among different models, which helps improve robustness. To assess the diversity, we compute heatmaps of pairwise correlations between the absolute reconstruction errors for each sensor across the different models. A high correlation indicates that two models behave similarly, while low correlation suggests complementary perspectives.

Figure 3a presents a heatmap of correlations of reconstruction errors, illustrating the degree of redundancy or complementarity between the ensemble models. The results confirm that while some models (e.g., RNN and DCRNN) exhibit similar behavior, others (e.g., DCRNN and GWN) capture distinct features, reinforcing the benefit of a diverse ensemble. Figure 3b provides more detail showing the correlations between predictions made by different model combinations for all sensors of the automation system and one of the two engines. Certain sensors such as the cruiser's heel or the level of tanks cannot be inferred from other sensors and is thus harder to predict for any model. However, there are also sensors which are handled differently only by some models.

4.1.1 Comparison with HBOS Baseline

To demonstrate the advantages of the (spatio)temporal ensemble approach, we compare its anomaly scores against a baseline method, namely Histogram-Based Outlier Score (HBOS) [12]. HBOS operates on single time steps and is based on the principle that anomalies often have low probability density in feature distributions. The method is very compute-efficient but in our use case it suffers from the fact that transient rapid changes (e.g. engine starts) are likely to get a high anomaly score as they occur much less frequently than steady states (constant speeds, idling).



(a) HBOS (b) Ensemble

Figure 4: Comparison of anomaly scores between the HBOS baseline and the ensemble method.

Figure 4 shows the histogram of anomaly scores from both HBOS and the proposed ensemble method. The HBOS distribution is separated into two parts with the engines running or not, whereas the ensemble captures more subtle, sustained deviations indicative of meaningful anomalies in a system with non-stationarity.

4.1.2 Qualitative Examples of Detected Anomalies

To provide a view of how anomalies manifest in the search-and-rescue cruiser CPS data, Figures 5 and 6 present examples of detected anomalies within a two-minute time window around an anomalous time step. We visualize all 110 signals to capture the system-wide behavior and interdependencies that are critical for understanding anomalies in a non-stationary, high-dimensional environment. Figure 5 illustrates a point anomaly, where a sudden current spike in the electrical system occurs while the cruiser is docked. The anomaly is reflected across multiple related sensors, reinforcing that even localized faults impact broader system behavior.

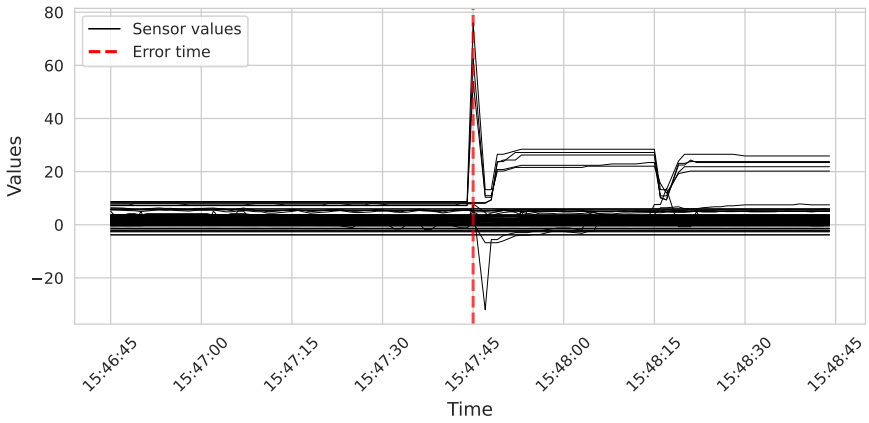


Figure 5: Example of detected point anomaly.

Figure 6 demonstrates a collective anomaly, where several sensors exhibit simultaneous drifts or oscillations that, while within nominal operating ranges individually, collectively indicate an unusual operating state. This highlights the advantage of a system-wide detection approach over single-signal methods, as anomalies may emerge from multi-sensor correlations rather than discrete threshold violations.

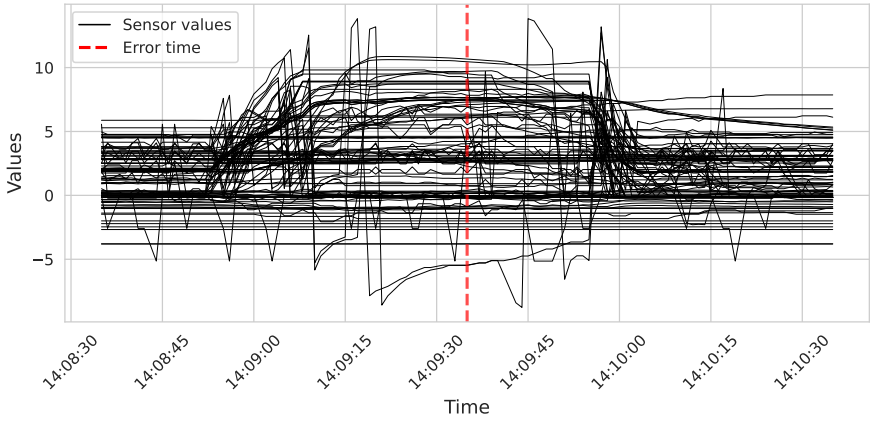


Figure 6: Example of detected collective anomaly.

By displaying all sensors, we emphasize the holistic nature of CPS monitoring, where anomalies cannot be interpreted in isolation. Interdependencies and the necessity of considering global context presents a challenge not only for anomaly detection models, but also for the user.

4.2 Stage 2: Anomaly Explanation Evaluation

The explanation stage aims to alleviate the cognitive burden by summarizing detected anomalies in natural language, allowing users to quickly interpret system deviations without the need for extensive manual inspection. Evaluating LLM performance requires assessing both the correctness and format of the generated outputs. We evaluate LLM performance across two models and two prompt structure options. Each configuration is tested on 10 anomalies.

Prompt Option	Input Tokens (K)	Model	Anomaly Hallucinations	Value Hallucinations	Format Adherence
Context	102.4	`4o-mini`	100%	100%	0%
Context	102.4	`o1-mini`	100%	100%	60%
Focused	3.9	`4o-mini`	100%	0%	0%
Focused	3.9	`o1-mini`	70%	0%	30%

Table 2: Percentage of detected issues for each LLM and prompt option.

The percentages in Table 2 represent the proportion of the 10 evaluated anomalies where each issue was observed for the given prompt option and model. For example, a 70% value under "Anomaly Hallucinations" means that in 7 out of 10 test cases, the LLM falsely identified at least one signal as anomalous when it was operating normally. Similarly, "Value Hallucinations" indicates cases where the LLM generated incorrect numerical values that were not present in the input data. "Format Adherence" measures the percentage of completions where the LLM strictly followed the structured response format defined in the prompt. A lower adherence percentage suggests that the model occasionally deviated from the required output structure, making automatic parsing more difficult.

We observe that hallucination rates are higher for *Context* prompts, likely due to the greater amount of data (tokens) included in the prompt and the models having to combine information from different tables. Adherence issues are more prevalent in `o1-mini`, which exhibits stronger

reasoning capabilities but also a higher tendency to deviate from instructions. These results reinforce the importance of structured outputs beyond structured prompts. Even when models receive well-formatted inputs, deviations in output format can make automatic parsing unreliable and disrupt user expectations. Ensuring strict adherence allows for a more predictable and user-friendly anomaly explanation system.

For *Context* prompts, both models frequently fabricate values or confuse scaled and unscaled representations. Given *Context* `o1-mini` attempts to infer root causes but fails to establish relationships between anomalies, instead attributing each sensor deviation to an independent issue. As for the *Focused* option, `4o-mini` often marks normal sensors as anomalous, seemingly influenced by the structured format, while `o1-mini` occasionally successfully identifies overarching failure patterns instead of listing multiple isolated issues.

These findings highlight the challenge of balancing detail and conciseness in structured prompts to ensure reliability while minimizing hallucinations. To further illustrate LLM failure cases, we present parts of outputs as examples highlighting specific issues.

Example: Value Hallucination (Model: `4o-mini` + *Context* prompt)

Generated Output: *"`PowerLargeAuxiliaryEngine` indicates a power output of 350.0 kW, which exceeds the upper limit of the expected normal range."*

Issue: The actual power is $\approx$ 30 kW. A fan is running at 350 rotations per minute.

Example: Anomaly Hallucination (Model: `o1-mini` + *Focused* prompt)

Generated Output: *"`EngineDesiredOperatingSpeed` shows a desired operating speed of 2305.0. This suggests the engine is pushed beyond its intended speed."*

Issue: The anomaly consists of an unusual combination of values, not exceeding normal ranges.

These results highlight the challenges of ensuring LLM reliability and reinforce the importance of structured prompts and data curation in preventing misinterpretations.

5 Discussion

Our two-stage anomaly detection framework effectively identifies anomalies using diverse temporal and spatiotemporal models while generating natural language explanations via LLMs. However, LLMs face challenges such as hallucination and inconsistent adherence to structured outputs. The self-supervised ensemble detects anomalies without labeled data, leveraging diverse models to improve generalization and reduce overfitting. It outperforms the HBOS baseline by capturing temporal dependencies and system context, detecting both point and collective anomalies, and offering complementary insights that reduce redundancy.

While LLMs enhance interpretability, they introduce issues such as hallucinations, incorrect or fabricated explanations, deviations from structured outputs, and trade-offs between explanation detail and correctness. Addressing these limitations requires improvements in prompt engineering and domain-specific fine-tuning. Our structured division of labor ensures that models handle high-dimensional data efficiently, LLMs generate human-readable explanations, data scientists optimize models and prompts, and users validate outputs against operational knowledge. This framework balances automation with interpretability, reducing cognitive load for users.

6 Conclusion and Future Work

We propose a two-stage anomaly detection framework integrating self-supervised learning and LLM explanations to improve detection and interpretability. The framework offers robust anomaly detection via a self-supervised ensemble, structured LLM-based explanations, and a defined division of labor for scalable, user-friendly anomaly detection. Future work involves

reducing LLM hallucinations through fine-tuned models and validation mechanisms, improving anomaly scoring with optimized ensemble weighting and domain-specific priors, refining models iteratively based on user feedback, and enabling real-time processing for CPS deployment.

Several enhancements could further improve the system. Retrieval-Augmented Generation [21] could provide additional context by retrieving relevant examples of normal and anomalous samples, improving LLM contrastive reasoning. Ontologies could structure domain knowledge more effectively, offering better grounding for explanations [26]. Fine-tuning LLMs on domain-specific anomaly detection tasks through reinforcement learning from human feedback [31] would help mitigate hallucinations and improve adherence to structured formats. Providing explicit normal and anomalous samples during inference could aid LLMs in distinguishing expected deviations from true faults [25]. Expanding domain knowledge integration would enhance model reliability by grounding explanations in real-world constraints. Lastly, incorporating AI agents that dynamically interact with users and adapt explanations based on feedback could further improve usability [41]. By addressing these areas, we aim to bridge machine learning advancements with practical CPS applications, enhancing safety, reliability, and interpretability in mission-critical systems.

Acknowledgments

This research has been conducted as part of the project SmartShip which is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union – NextGenerationEU.

References

[1] M. Ahmed, A. N. Mahmood, and J. Hu. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19–31, 2016.

[2] S. Amershi, D. Weld, M. Vorvoreanu, A. Fourney, B. Nushi, P. Collisson, J. Suh, S. Iqbal, P. N. Bennett, K. Inkpen, et al. Guidelines for human-ai interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–13, 2019.

[3] J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga. Do deep neural networks contribute to multivariate time series anomaly detection? *Pattern Recognition*, 132:108945, 2022.

[4] J. L. Augustin and O. Niggemann. Self-supervised graph structure learning for cyber-physical systems. *IFAC-PapersOnLine*, 58(4):204–209, 2024.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[7] A. Cini and I. Marisca. Torch Spatiotemporal, 3 2022.

[8] A. Deng and B. Hooi. Graph neural network-based anomaly detection in multivariate time series. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 4027–4035, 2021.

[9] J. Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[11] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022.

[12] M. Goldstein and A. Dengel. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. *KI-2012: poster and demo track*, 1:59–63, 2012.

[13] S. Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[14] A. Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain informatics*, 3(2):119–131, 2016.

[15] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 387–395, 2018.

[16] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

[17] J. Jiang, F. Wang, J. Shen, S. Kim, and S. Kim. A survey on large language models for code generation. *arXiv preprint arXiv:2406.00515*, 2024.

[18] H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. Wallach, and J. Wortman Vaughan. Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–14, 2020.

[19] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[20] J. Lee, B. Bagheri, and H.-A. Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015.

[21] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[22] Y. Li, R. Yu, C. Shahabi, and Y. Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[23] Z. Li, Z. Rao, L. Pan, P. Wang, and Z. Xu. Ti-mae: Self-supervised masked time series autoencoders. *arXiv preprint arXiv:2301.08871*, 2023.

[24] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

[25] J. Liu, C. Zhang, J. Qian, M. Ma, S. Qin, C. Bansal, Q. Lin, S. Rajmohan, and D. Zhang. Large language models can deliver accurate and interpretable time series anomaly detection. *arXiv preprint arXiv:2405.15370*, 2024.

[26] B. Ludwig, A. Diedrich, and O. Niggemann. Using ontologies to create logical system descriptions for fault diagnosis. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2024.

[27] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *Esann*, volume 2015, page 89, 2015.

[28] O. Niggemann, B. Zimmering, H. Steude, J. L. Augustin, A. Windmann, and S. Multaheb. Machine learning for cyber-physical systems. In *Digital Transformation: Core Technologies and Emerging Topics from a Computer Science Perspective*, pages 415–446. Springer, 2023.

[29] OpenAI. Gpt-4o-mini-2024-07-18. Accessed via OpenAI API, July 2024. Available at: `https://platform.openai.com`.

[30] OpenAI. o1-mini-2024-09-12. Accessed via OpenAI API, September 2024. Available at: `https://platform.openai.com`.

[31] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[32] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[33] L. Reynolds and K. McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 CHI conference on human factors in computing systems*, pages 1–7, 2021.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[35] R. Sennrich. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[36] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani, H. Cole-Lewis, S. Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

[37] A. Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[38] D. Wang, Q. Yang, A. Abdul, and B. Y. Lim. Designing theory-driven user-centric explainable ai. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–15, 2019.

[39] A. Windmann, H. Steude, and O. Niggemann. Robustness and generalization performance of deep learning models on cyber-physical systems: A comparative study. *arXiv preprint arXiv:2306.07737*, 2023.

[40] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang. Graph wavenet for deep spatial-temporal graph modeling. *arXiv preprint arXiv:1906.00121*, 2019.

[41] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

[42] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi. Deep learning for time series anomaly detection: A survey. *ACM Computing Surveys*, 57(1):1–42, 2024.

[43] K. Zhang, Q. Wen, C. Zhang, R. Cai, M. Jin, Y. Liu, J. Y. Zhang, Y. Liang, G. Pang, D. Song, et al. Self-supervised learning for time series analysis: Taxonomy, progress, and prospects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[44] H. Zhao, Y. Wang, J. Duan, C. Huang, D. Cao, Y. Tong, B. Xu, J. Bai, J. Tong, and Q. Zhang. Multivariate time-series anomaly detection via graph attention network. In *2020 IEEE international conference on data mining (ICDM)*, pages 841–850. IEEE, 2020.

[45] Z. Zhou and R. Yu. Can llms understand time series anomalies? *arXiv preprint arXiv:2410.05440*, 2024.

TOWARDS ADAPTIVE TRAFFIC SIGNAL CONTROL THROUGH FOUNDATION MODELS AND REINFORCEMENT LEARNING

Lukas Klein, Arthur Müller, Magnus Redeker
Fraunhofer IOSB, IOSB-INA Lemgo
Fraunhofer Institute of Optronics, System Technologies and Image Exploitation
lukas.klein@iosb-ina.fraunhofer.de

ABSTRACT

Traffic Signal Control (TSC) is pivotal for managing urban traffic flow and enhancing intersection safety. Traditional TSC systems are rule-based and tailored to specific intersections, requiring substantial training and resources, which restricts their flexibility. This paper proposes a novel adaptive, scalable solution utilizing Foundation Models (FM) and Reinforcement Learning (RL), designed to handle diverse urban intersections efficiently without extensive retraining. The approach leverages advanced neural network architectures, including attention mechanisms, to improve generalization capabilities across different intersection topologies. A safety control mechanism aligned with traffic regulations ensures the safe operation of traffic signals, significantly enhancing the system's reliability. By systematically classifying intersection types, the method tailors the control strategies to specific traffic scenarios, further reducing implementation times and expertise requirements. This FM- and RL-based approach not only reduces resource demands but also promises more efficient traffic flow and improved safety in various urban settings.

Keywords Traffic Signal Control · Foundation Models · Reinforcement Learning · Rapid Application in Heterogeneous Environments · Safe and Adaptive Traffic Flow Optimization

1 Introduction

The optimization of traffic flow is essential for ensuring the efficient utilization of urban road networks, reducing congestion, minimizing travel delays, and improving overall traffic system performance. Traffic signal control (TSC) systems face several challenges, including the integration of modes of transport, which involves considering the needs of pedestrians, cyclists, cars and truck drivers, and public transport to create a comprehensive traffic management system. The escalating traffic volumes and resulting congestion significantly extend waiting times for commuters and the logistics industry, thereby escalating operational costs and reducing overall efficiency. Safety aspects are crucial, focusing on ensuring the safety of both pedestrians and drivers to minimize the risk of accidents at intersections.

Reinforcement learning (RL) presents a promising AI-driven approach for optimizing TSC by leveraging vast amounts of real-time data. Advances in traffic sensing technologies, such as high-resolution cameras, radar sensors with a high range and accuracy, and the associated access to high-quality data provided by vehicle-to-everything (V2X) technologies, enable data at scale, facilitating the development of more adaptive and efficient RL-based TSC.

RL and other Artificial Intelligence (AI)-based techniques for TSC have already been extensively studied [1, 2, 3] and have even been evaluated in real-world settings, including Cologne,

Essenbach, Hamm, Ingolstadt, and Lemgo in Germany [4, 5]. However, challenges remain, particularly in terms of generalizability since most existing approaches require specific training of RL-agents for each individual intersection, leading to high computational costs, long implementation times, and a demand for specialized AI and traffic engineering expertise — resources that many municipalities, traffic engineering firms, and research institutions lack.

In order for AI-based TSC optimization to gain widespread adaptation, it must not only match or exceed the performance of traditional rule-based methods but also reduce implementation complexity and expertise requirements. To address these challenges, this paper introduces a Foundation Model (FM)-based approach that enhances adaptability, scalability, and efficiency. Key aspects of this approach include *topology-agnostic adaptability*, enabling the model to manage diverse intersections types without extensive retraining, and the creation of a *robust training dataset* that integrates both synthetic and real-world traffic data. By leveraging these advancements, the proposed method aims to provide a more flexible, cost-effective, and broadly applicable solution for modern traffic management.

Beyond optimizing traffic flow, ensuring safety remains a crucial aspect of AI-driven TSC. Implementing safety control mechanisms that comply with traffic regulations is necessary to guarantee the safe operation of traffic signals and prevent hazardous situations at intersections. Furthermore, a *systematic classification of intersection types* based on relevant characteristics will facilitate a deeper understanding and modeling of diverse traffic scenarios, ultimately contributing to more effective traffic management solutions. Additionally, leveraging *advanced neural network architectures*, such as attention mechanisms, can enhance the performance and generalization capabilities of the RL-based system, allowing it to better adapt to new and unseen traffic conditions and intersection topologies.

By addressing these aspects, AI-driven TSC solutions will become more efficient, adaptable, and accessible, ultimately improving urban traffic management systems.

This paper is structured as follows: Section 2 provides an overview of the state-of-the-art in RL-based TSC. Section 3 conceptualizes the proposed foundation model approach. Finally, Section 4 discusses the applicability and generalizability of the results and outlines directions for future research and development.

2 Preliminaries

2.1 Traffic Signal Control

Traffic Signal Control (TSC) refers to the control of intersections within urban road networks to manage vehicular and pedestrian movement. Conventional TSC policies, embedded in traffic controllers, are designed by traffic engineers and rely on rule-based mechanisms to govern the operation of traffic signals. These control strategies are dynamic and adjust based on real-time traffic conditions, which are detected through various sensing technologies such as cameras, inductive loop detectors, and pedestrian buttons.

Traffic signals (or *traffic lights*) serve as visual signaling devices that provide guidance to road users by regulating the right-of-way for specific lanes or movements. Multiple traffic signals can be grouped into a *signal group*, which is defined as a set of traffic signals that control specific traffic flows simultaneously. This grouping ensures synchronized signaling for particular directions or lanes, optimizing traffic flow and enhancing intersection efficiency.

Traffic phases represent a specific combination of traffic signal states that define a stable traffic light configuration, during which the signal status remains unchanged. Each phase typically corresponds to a set of compatible traffic flows, such as vehicles traveling in the same or non-conflicting directions, ensuring smooth and coordinated movement through an intersection.

Phase transitions occur when the Traffic Light System (TLS) control mechanism initiates a shift from one traffic phase to another to regulate traffic flow effectively. These transitions follow a structured sequence, beginning with the traffic light switching from green to yellow and subsequently to red. Once a phase concludes, the corresponding signal groups of the next phase are activated by switching to green, ensuring a seamless and orderly traffic flow.

Phase-based control in traffic signal systems manages predefined combinations of signal groups as unified entities. These transitions between these phases are predetermined in control protocols and are specific to each intersection. Typically, only a subset of possible phase transitions is permitted to maintain a conflict-free and efficient operation. These transitions are often illustrated in *phase transition diagrams*, which outline the sequence of phase changes to uphold traffic safety. In contrast, *signal group-based control* operates individual signal groups independently, adjusting their timing separately while ensuring conflict-free traffic movement across the intersection.

In reinforcement learning (RL)-based TSC, it is essential to ensure that agents execute phase and signal group transitions in a manner that adheres to traffic safety regulations. Müller, Rangras, Ferfers, Hufen, Schreckenberg, Jasperneite, Schnittker, Waldmann, Friesen, and Wiering [5] propose a safety layer, a mechanism designed to evaluate and regulate the switching decisions made by RL agents. This layer allows phase transitions only if they meet predefined safety criteria, rejecting unsafe transitions through *action masking* to prevent violations of traffic regulations. The integration of such safety mechanisms can be implemented in various ways; therefore, the terms *safety layer* and *safety control system* are used to refer to any approach that enforces safety constraints within RL-based TSC.

2.2 Reinforcement Learning

Reinforcement learning (RL) agents interact with their environment with the objective of making optimal decisions and executing corresponding actions. At each time step t, an agent observes the current state of the environment, denoted as s_t. Based on this information, the agent selects the subsequent action a_t, to undertake within the environment. The agent's behavior is governed by a policy π, which defines a mapping from states to actions, expressed as $\pi(a_t|s_t)$. To assess the agent's performance in an RL framework, the environment provides a reward signal r_{t+1} which quantifies the effectiveness of the chosen action in achieving the specified objective. The fundamental goal of an RL agent is to maximize the expected cumulative reward over time.

This decision-making process is typically formalized using *Markov Decision Processes* (MDPs), a mathematical approach that simplifies sequential decision-making into a structured format. An MDP is defined as a tuple comprising the following elements [6]:

- **State space** S: A set of states representing the environment's current condition.
- **Action space** A: A set of possible actions from which the agent selects one at each time step t.
- **State transition function** $T(s_t, a_t, s_{t+1})$: A probability distribution over the successor state s_{t+1}, given the current state s_t and action a_t.
- **Reward function** $R(s_t, a_t)$: A function that assigns a reward r_{t+1} to the agent based on the action a_t taken in state s_t.
- **Discount factor** $\gamma \in [0, 1]$: A parameter that determines the importance of future rewards, with γ close to zero prioritizing immediate rewards.

A key property of an MDP is the *Markov property*, which states that the current state s_t contains all relevant information about the past states and actions, meaning that the successor state s_{t+1} depends solely on the current state s_t and the action a_t, independent of prior states and actions. MDPs can model both *continuous tasks*, which have no predefined endpoint, and *episodic tasks*, which terminate upon reaching a specific state. Consequently, MDPs can address a diverse array of decision-making scenarios within the RL domain.

2.3 Foundation Models

Foundation Models (FMs) are large-scale, pre-trained neural networks designed to serve as adaptable platforms for a wide range of downstream tasks across various domains, including natural language processing (NLP), computer vision, and beyond. Prominent examples of FMs include models such as BERT [7], GPT [8], and Llama [9], which utilize vast amounts of data and advanced architectures to learn complex patterns and relationships inherent in the data.

The primary advantage of FMs is their ability to generalize across diverse tasks, significantly reducing the need for task-specific training. This generalization is facilitated through *transfer learning*, a paradigm in which models pre-trained on large and varied datasets are fine-tuned for specific applications [10]. FMs are typically built on the *transformer architecture* [11], which employs *self-attention mechanisms* to assign different weights to input tokens of varying lengths, thereby enabling the model to capture contextual relationships more effectively.

While transformer-based models generally excel at processing sequential data, it is important to note that, in the context of TSC, token sequences are not directly processed. Instead, the attention mechanism remains highly beneficial as it assists the model in identifying which elements within the input space correlate with specific traffic phases. Essentially, it calculates the relevance of various inputs (such as traffic flow data or sensor readings) to different phases of signal control, allowing the model to focus on the most pertinent information for each phase transition. Additionally, attention mechanisms facilitate massive parallelization during training, enabling FMs with transformer architecture to converge faster and overall improve their performance on large datasets.

2.4 Related Work

Several promising results are evident from research activities in TSC optimization:

- Ault and Sharon [12] present a toolkit for the development and comparison of various RL-based TSC systems and the benchmarking of scenarios based on realistic traffic conditions from the German cities of Cologne and Ingolstadt. While this work offers valuable insights, it primarily focuses on the training of only two.
- Zhao, Dai, Chen, Lin, Lv, and Wang [13] address the Sim2Real challenge by employing descriptive learning with a Wasserstein GAN to generate different traffic scenarios, utilizing transformer-based learning to discern traffic dynamics, and fine-tuning their FM through real-world applications. However, not addressed are structural and spatial features of intersections, which could provide valuable information for RL learning.
- Zang, Yao, Zheng, Xu, Xu, and Li [14] develop a metamodel that requires adaptation to specific traffic signals. However, it is not structure-agnostic, as it is specifically designed for 4-arm intersection topologies only. The approach is based on basic model-agnostic meta-learning (MAML), instead of improved and more robust later variants [15, 16]. Additionally, an outdated convolutional neural network (CNN) architecture is employed and not yet the current de facto standard of attention mechanisms in FMs.
- Zhang, Liu, Zhang, Zheng, and Yu [17] seek to enhance the generalization of TSC by training on diverse traffic flow data generated via a Wasserstein generative adversarial network (WGAN). Their approach integrates flow clustering with model-agnostic meta-learning to effectively adapt to various traffic conditions, demonstrating good performance across different traffic flows. However, the topology of the simulation models remains constant, which may restrict the FMs ability to generalize across topologically different intersections.
- Oroojlooy, Nazari, Hajinezhad, and Silva [18] propose a universal attention-based TSC-model that utilizes attention mechanisms to generalize across various intersection topologies. However, it lacks a dedicated safety mechanism to ensure only permissible actions are taken, which is crucial for a future deployment in real-world scenarios.

On the bottom line, research should focus on generalization capabilities across intersection types topology-agnostic TSC adaptability applying advanced neural network architectures, targeting straightforward effort-less and cost-effective TSC optimization.

3 Towards Adaptive Traffic Signal Control through Foundation Models and Reinforcement Learning

This section outlines the methodology for developing an FM for RL-based TSC, focusing on the integration of synthetic and real-world data to create a robust and adaptable dataset as depicted in Figure 1. By leveraging advanced techniques such as attention mechanisms and

systematic classification of intersection types, the approach aims to facilitate dynamic traffic signal management across diverse urban environments. The outlined framework not only targets the effective training of RL agents but also ensures compliance with essential safety regulations, ultimately contributing to more efficient, adaptive urban traffic systems.

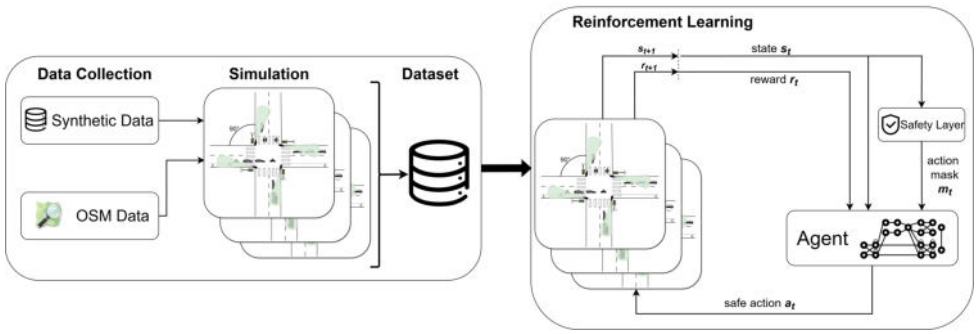


Figure 1: High-Level illustration of the data collection process and reinforcement learning workflow for training a Foundation Model with action masking. OpenStreetMap-Logo: © OpenStreetMap contributors, CC-BY-SA 2.0.

3.1 Core Conception Principals

The following subsections describe the core design principles. These include data collection and preparation, the simulation framework, safety mechanisms, and ensuring agnosticism towards intersection and environment topology.

3.1.1 Data Collection and Preparation

In order to create a diversified dataset, simulation models of individual intersections are created using both synthetically generated networks and real-world data from OpenStreetMap (OSM, [19]). The generation of synthetic traffic networks enables the mapping of a variety of traffic scenarios and conditions, thereby providing a valuable basis for training an FM. The generation of networks based on real intersection situations from OSM enriches the dataset with real-world realistic data. The combination of synthetic and real data facilitates the generalization of the developed FM.

Furthermore, the dataset is augmented with multiple phase diagrams for each specific intersection, enabling the mapping of diverse switching logics for the same topology. These phase diagrams are generated using a tool to be developed from the respective network topology and traffic engineering expert knowledge in order to automatically generate a safety control system. Such a safety control system represents a pivotal component in the outlined methodology, and is instrumental in training of the FM to comply with the mandatory safety regulations governing the switching of traffic lights in the field.

To ensure that the dataset can be employed as a benchmark for the training and evaluation of RL for TSC, it is essential to classify different isolated intersection types based on relevant characteristics such as topology, participant groups (e.g., motorized and non-motorized traffic), public transport, traffic patterns (e.g., partial load and full load), and speed limits. The objective is to develop a systematic description of the various intersection types, which will aid in understanding of intersection design and serve as a foundation for the synthetic generation of simulation models for the dataset. To align the dataset with the actual distribution of intersections in the real world, a minimum number of intersections per type will be selected for inclusion.

3.1.2 Simulation Framework

A systematic classification of intersection types will be undertaken based on their topological characteristics as described in OpenStreetMap and informed by expert knowledge. Subsequently, the OSM Web Wizard, a tool developed by the *Deutsches Zentrum für Luft- und Raumfahrt*

(DLR), will be employed to generate a simulation scenario in the microscopic traffic simulation tool SUMO based on a selection of an OSM map section [20, 21] as in Figure 2. This will be followed by a post-processing phase with the objective of refining the resulting scenarios. The synthetic simulation models will then be generated to correspond with the identified intersection types, covering a range of traffic volumes, specifically full-load, partial-load and light-load, with calibration aimed at full-load conditions.

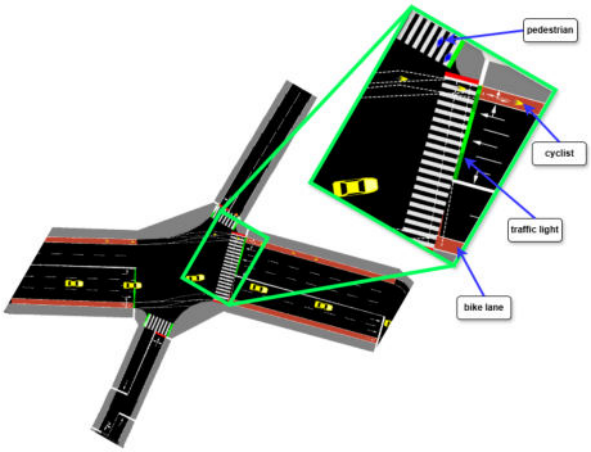


Figure 2: Illustration of an intersection simulated in SUMO, highlighting key traffic elements such as pedestrians, cyclists, bike lanes, and traffic lights.

In addition, different groups of road users and public transport modalities will be defined for each intersection to ensure a comprehensive and representative simulation framework. Each simulation model per intersection will also be equipped with a safety control mechanism, referred to as the *safety layer*, which will adhere strictly to the most essential guidelines for the design and operation of traffic signal systems in Germany [22].

To enhance efficiency, the interim time matrices and corresponding blocking times are automatically generated. As the interim times depend on the number of traffic streams at an intersection and the lengths of the entry and exit paths, it is possible to automate the calculation of these interim times based on the intersection's topology. The interim time matrix provides a representation of all interim times at a given intersection, facilitating comparison and reading across all traffic streams. In the domain of traffic engineering, the term *interim time* refers to the duration that must elapse between the end of the green time of one traffic stream and the start of the green phase of another, partially or fully conflicting, incoming traffic stream and are a crucial safety-related aspect, as they ensure that incoming and outgoing traffic flows can be switched without conflict, thus preventing potential collision scenarios.

3.1.3 Safety Mechanisms

The safety layer will be implemented natively in Python, eliminating the need for additional traffic engineering workstations such as *LISA*, [23]. This will ensure accessibility to a wider range of user groups while improving overall performance.

3.1.4 Intersection and Environment Topology Agnosticity

Previous studies on reinforcement learning for traffic signal control have primarily focused on phase-based switching, where RL agents operate within predefined traffic phases and phase transition plans. While this approach has demonstrated feasibility, it limits the system's adaptability to dynamic traffic conditions. To enable a more flexible and responsive control strategy, ongoing research aims to develop a Foundation Model (FM) that supports both phase-based and signal group-based switching, allowing for more granular and adaptive traffic signal management.

In order to achieve this, specific phase diagrams must be automatically generated for each intersection. A comprehensive topological analysis must be conducted, extracting key structural characteristics such as the number of arms, available lanes, and prevailing traffic flows. This analysis enables the modeling of traffic movement and the determination of the minimum number of required phases. By systematically analyzing potential conflicts between these phases, a basic phase sequence diagram can be generated while ensuring compliance with essential safety regulations.

For signal group-based switching, a safety control layer is required to prevent RL agents from activating conflicting signal groups simultaneously. This layer ensures that only non-conflicting traffic movements are permitted, thereby maintaining safe and efficient traffic operations.

To integrate RL-based control within a simulation environment, an environment class is designed to facilitate seamless interaction between RL agents and the traffic simulation. Built upon the Gymnasium framework [24], this implementation allows for the creation of environment instances that reflect specific intersection topologies, traffic volumes, public transportation modalities, and user groups. Users can define scenario parameters, enabling the automated instantiation of tailored environments from a pre-existing dataset. Each unique parameter combination results in a corresponding environment instance that accurately represents the defined traffic scenario.

Furthermore, the environment class provides a standardized interface for interacting with simulation components while incorporating the safety control layer to enhance reliability and ensure regulatory compliance. This design promotes modularity, making it easier to integrate RL agents with diverse traffic environments and facilitating the extensibility of RL frameworks, such as Ray RLlib. By enabling structured and safe RL training, this approach supports the development of scalable, real-world-applicable AI-driven traffic signal control systems.

3.2 Leveraging Reinforcement Learning and Foundation Models in the Concept of Adaptive Traffic Signal Control

To enable low-effort, cost-effective, and adaptive TSC, the development of intersection topology-agnostic neural network architectures is a critical focus of research. A topology-agnostic approach ensures that the same model architecture can flexibly adapt to different intersection types without requiring structural modifications or retraining for each new scenario. This adaptability extends not only to dynamic input representations that reflect the unique characteristics of each intersection but also to variable output structures, corresponding to the number of signal groups or predefined traffic phases at a given location.

To achieve this level of generalizability, advanced techniques such as attention mechanisms and Graph Neural Networks (GNNs) play a crucial role in RL-based TSC. GNNs are particularly well-suited for modeling traffic networks, where lanes, traffic movements, and intersections can be represented as nodes, and their relationships—such as connectivity, priority rules, and turning restrictions—can be represented as edges. Unlike traditional neural networks, GNNs allow flexible representations of varying traffic intersection topologies, making them an ideal choice for RL-based adaptive signal control.

Additionally, attention mechanisms further enhance the adaptability of RL-based TSC by dynamically weighting the relevance of different inputs within the graph. This is particularly useful in variable-sized inputs, where the model must determine which traffic flows, lanes, or intersections should be given priority based on real-time conditions. Unlike traditional feedforward architectures, which treat all input features equally, attention mechanisms enable the model to focus on the most relevant traffic features at any given moment, improving decision-making robustness.

To ensure effective integration into the RL framework, the design of these neural architectures must adhere to Markov Decision Process (MDP) principles, ensuring a well-structured formulation of states, observations, actions, and rewards. A key component of this structure is the incorporation of historical traffic data, allowing RL agents to learn from past traffic patterns rather than relying solely on instantaneous states. By recognizing temporal dependencies, agents can anticipate congestion patterns and peak-hour trends, enabling more proactive and adaptive

traffic signal control. By combining GNN-based representations with attention mechanisms, RL policies can flexibly handle input and output variations, enhancing the scalability and efficiency of AI-driven TSC Foundation Models.

3.2.1 State space

In RL, the state space is defined as a set encompassing all potential states of the environment. At each time step t during the RL training process, a state s_t is provided as the input to the training framework's policy. Generally, this state represented as a vector containing information about the environment's current state, such as the current phase of the traffic lights at the intersection or the queue length on each lane.

To facilitate the learning of a foundation model with the greatest possible breadth of information, it is essential to construct the state space meticulously to ensure its generalizability across different intersection topologies. The agent requires an awareness of the current signal phase and the average waiting time of road users at each incoming lane to accurately assess the prevailing traffic conditions and make optimal decisions to improve traffic flow. Additionally, to enhance generalization capabilities, it is essential to differentiate between varying traffic load levels (e.g., full load, partial load) and incorporate temporal information, such as the duration of the current signal phase or the average lane occupancy over a predefined time window (e.g., the last ten minutes). Beyond traffic dynamics and temporal information, capturing spatial information is fundamental, ensuring the RL agent learns the topological structure of an intersection. The agent must recognize key attributes of the intersection, such as the number of lanes, the number of arms, and the topological layout (e.g, the angles between lanes). This enables the agent to tailor its decision based on the structural characteristics of the traffic environment, ensuring optimized TSC strategies across varying intersection designs.

For a fixed intersection topology, the state space can be represented as a fixed-length vector, simplifying neural network training by ensuring a uniform input size at each time step t. However, this approach becomes problematic when training an FM across multiple intersections, as input size variations arise from differences in intersection structures. Therefore, it is crucial to develop and utilize an appropriate policy architecture, such as an artificial neural network, that accommodates variable input sizes.

A promising approach to address variable-size inputs is to define a fixed-length vector that applies for each lane at an intersection. This vector encapsulates all the relevant spatial, temporal, and contextual information for that particular lane. In cases where certain lanes provide less information, zero-padding can be applied to maintain a uniform vector size, enabling an information-size-agnostic representation. To further enhance the topology-agnostic nature of the state space, the entire intersection state can be modeled as a graph, where nodes represent lanes and edges capture the logical dependencies between lanes (e.g., connectivity, conflicting movements, or priority rules). This graph-based representation allows the RL agent to adapt dynamically to different intersection layouts while preserving crucial spatial relationships, ultimately improving its ability to generalize and optimize traffic signal control across diverse urban environments.

3.2.2 Action space

Analogous to the state space, the action space is defined as a set of all possible actions that an agent can perform in the environment, serving as the output of the policy during the RL process. When training an RL agent for a specific intersection, the output layer is typically designed with a fixed dimension corresponding to the number of switchable phases or signal groups, thereby defining a discrete action space. The logits—i.e., the probability distribution over possible actions—are then sampled to determine the next action.

When training an FM across different intersections, the output size of the policy varies based on the number of defined phases or signal groups. To achieve output agnosticism, several techniques can be employed. One straightforward approach involves dynamically adjusting the output layer to match the action space size of each intersection during training. However, this approach necessitates fine-tuning for each new intersection. A more scalable and generalizable

approach leverages attention mechanisms. Here, an embedding representation is computed for each lane-specific feature vector, capturing relevant lane-specific information. An attention mechanism then assigns weights to these embeddings, determining the policy's output. A downstream translation mechanism subsequently maps the attention score of an embedding back to their corresponding lanes, associating them with the correct signal group or phase.

Furthermore, the security layer not only verifies the switching intentions of the agents to ensure compliance with legal and safety regulations, but also indicates which phases or signal groups can be switched in the subsequent time step. Following [25], a masking vector is generated from these permitted phases or signal groups, constraining the action space of the agents and enabling only allowed phase or signal group transitions. This action masking technique must be adapted to the network architecture implemented.

3.2.3 Reward function

The formulation of a reward function aimed at optimizing traffic flow is a crucial step in enabling a foundation model to make well-informed TSC decisions. This process is guided by the *Handbuch für die Bemessung von Straßenverkehrsanlagen* (HBS) [26], which provides standardized metrics for the evaluation of traffic flow efficiency. The proposed reward function may serve as a benchmarking metric, leveraging the concept of *Level-of-Service* (LOS) as a foundational element. LOS is a discrete metric that assesses roadway segments and traffic directions based on the average waiting time of road users in seconds. It differentiates between motorized and non-motorized users, enabling the weighting of various groups of traffic participants. The overall LOS of an intersection is determined by the worst-performing traffic direction, ensuring that congestion in any direction is appropriately penalized. By incorporating LOS into the reward function, the FM can be trained to prioritize traffic efficiency while considering the needs of all road users.

A crucial consideration in the design of a suitable reward function involves addressing the temporal constraints of real-world traffic systems. In a naive simulation of a TSC, the agent receives an observation and reward at each time step t and can execute an action at any given moment, resulting in an equidistant interaction between the agent and the environment. However, in real-world traffic systems, actions cannot be executed at arbitrary time steps due to regulatory constraints such as minimum green times and mandatory phase transitions. Consequently, the intervals between successive decisions of RL-agent's are not equidistant, meaning that the environment continues to evolve even when the RL agent cannot take an action. To address this, it is essential to employ an appropriate reward aggregation strategy, ensuring that all rewards accumulated during non-decision periods are incorporated into the learning process rather than being lost. This approach guarantees that the agent accurately evaluates the impact of its actions over time.

Furthermore, to enhance the adaptability of the FM, its architectural framework must support signal group-based control, allowing agents to execute simultaneous actions that modify the states of multiple signal groups concurrently. To validate these concepts, a proof-of-concept prototype will be developed and trained on a subset of the dataset. This functional validation phase is crucial for assessing the model's applicability across diverse intersection topologies, demonstrating its feasibility and effectiveness in real-world urban traffic management. By integrating advanced AI techniques with well-established traffic engineering principles, this approach aims to enhance the efficiency, adaptability, and robustness of next-generation RL-based TSC systems.

4 Conclusion and Outlook

This paper introduced the concept of a Foundation Model (FM) for Reinforcement Learning (RL)-based Traffic Signal Control (TSC), designed to generalize across diverse intersection topologies by integrating attention mechanisms and graph-based representations. By addressing the challenge of variable input and output sizes, the proposed FM eliminates the need for training individual models for each intersection, significantly reducing computational demands and implementation effort. The topology-agnostic approach, supported by techniques such as

Graph Neural Networks and attention mechanisms, ensures that the model can dynamically adapt to different intersection structures, traffic patterns, and regulatory constraints.

A key contribution of this work will be the development of a benchmark dataset, derived from a combination of synthetic and real-world traffic networks, sourced from OpenStreetMap (OSM). This dataset will facilitate the training, evaluation, and comparison of RL-based TSC methods in a standardized manner, enabling municipalities, traffic engineers, and researchers to implement AI-driven traffic management solutions without requiring extensive AI and traffic engineering expertise or substantial computational resources. Furthermore, the integration of traffic simulations allows for realistic scenario modeling, enhancing the applicability and robustness of the proposed approach.

To ensure compliance with traffic regulations and safety standards, the model incorporates a safety control mechanism, extending the work of Müller, Rangras, Ferfers, Hufen, Schreckenberg, Jasperneite, Schnittker, Waldmann, Friesen, and Wiering [5], which enforces constraints on signal group switching and prevents unsafe phase transitions. This regulatory framework ensures that RL-based TSC solutions remain legally and functionally viable for real-world deployment.

The FM proposed in this paper lays the foundation for future research, focusing on:

- Extending the dataset to cover a broader range of urban environments and intersection types, ensuring greater generalization and real-world applicability.
- Refining the neural network architectures, particularly in optimizing the integration of adaptive mechanisms, such as attention mechanisms and graph-based representations, to enable RL agents to efficiently process and respond to different intersection topologies without requiring intersection-specific retraining.
- Validating the model in large-scale simulations and real-world deployments, bridging the gap between theoretical advancements and practical AI-based traffic management.

Making the benchmark dataset open-source will enable ongoing improvements in RL-based TSC by providing researchers with a standardized, accessible platform for testing and development. The targeted generalizing FM will significantly reduce implementation time, removing the need for costly, intersection-specific retraining, thereby democratizing access to AI-driven traffic optimization for municipalities, researchers, and companies with or without limited resources.

Acknowledgment

This work is part of the KISLEK project (19F1185A) and was supported by the German Federal Ministry for Digital and Transport (BMDV). The authors are responsible for the content.

References

[1] W. Miao, L. Li, and Z. Wang. “A Survey on Deep Reinforcement Learning for Traffic Signal Control”. In: *33rd IEEE CCDC*. IEEE.

[2] K.-L. A. Yau. “A survey on reinforcement learning models and algorithms for traffic signal control”. In: *ACM Computing Surveys* (2017).

[3] H. Wei, G. Zheng, V. Gayah, and Z. Li. “Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation”. In: *ACM SIGKDD Explorations Newsletter* 22.2 (2021).

[4] ZEIT ONLINE GmbH. *Wie KI-Ampeln Fußgänger bevorzugen*. URL: `https://www.zeit.de/mobilitaet/2024-12/strassenverkehr-ampeln-kuenstliche-intelligenz-fussgaenger-sicherheit/komplettansicht`.

[5] A. Müller, V. Rangras, T. Ferfers, F. Hufen, L. Schreckenberg, J. Jasperneite, G. Schnittker, M. Waldmann, M. Friesen, and M. Wiering. “Towards Real-World Deployment of Reinforcement Learning for Traffic Signal Control”. In: *20th IEEE ICMLA*. 2022. URL: `https://10.1109/icmla52953.2021.00085`.

[6] R. S. Sutton. “Reinforcement learning: An introduction”. In: *A Bradford Book* (2018).

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: 10.48550/ARXIV.1810.04805. URL: https://arxiv.org/abs/1810.04805.

[8] OpenAI et al. *GPT-4 Technical Report*. 2023. DOI: 10.48550/ARXIV.2303.08774. URL: https://arxiv.org/abs/2303.08774.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. *LLaMA: Open and Efficient Foundation Language Models*. 2023. DOI: 10.48550/ARXIV.2302.13971. URL: https://arxiv.org/abs/2302.13971.

[10] L. Torrey and J. Shavlik. “Transfer learning”. In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. “Attention Is All You Need.(Nips), 2017”. In: *arXiv preprint arXiv:1706.03762* 10 (2017), S0140525X16001837.

[12] J. Ault and G. Sharon. “Reinforcement learning benchmarks for traffic signal control”. In: *35th Neural Information Processing Systems Datasets and Benchmarks Track*. 2021.

[13] C. Zhao, X. Dai, Y. Chen, Y. Lin, Y. Lv, and F.-Y. Wang. “Parallel Learning Based Foundation Model for Networked Traffic Signal Control”. In: *26th IEEE ITSC*. 2023.

[14] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li. “Metalight: Value-based meta-reinforcement learning for traffic signal control”. In: *Proceedings of the AAAI conference on artificial intelligence*. 2020.

[15] C. Finn, P. Abbeel, and S. Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. 2017.

[16] J. Beck, R. Vuorio, E. Z. Liu, Z. Xiong, L. Zintgraf, C. Finn, and S. Whiteson. “A survey of meta-reinforcement learning”. In: *arXiv preprint arXiv:2301.08028* (2023).

[17] H. Zhang, C. Liu, W. Zhang, G. Zheng, and Y. Yu. “Generalight: Improving environment generalization of traffic signal control via meta reinforcement learning”. In: *29th ACM international conference on information & knowledge management*. 2020.

[18] A. Oroojlooy, M. Nazari, D. Hajinezhad, and J. Silva. “Attendlight: Universal attention-based reinforcement learning model for traffic signal control”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4079–4090.

[19] OpenStreetMap Foundation. *OpenStreetMap (OSM)*. URL: https://www.openstreetmap.org.

[20] Eclipse Foundation. *Eclipse Simulation of Urban MObility (SUMO)*. URL: https://eclipse.dev/sumo/.

[21] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. “Microscopic Traffic Simulation using SUMO”. In: *21st IEEE ITSC*. 2018. URL: https://elib.dlr.de/124092/.

[22] FGSV Verlag GmbH. *Richtlinien für Lichtsignalanlagen - Lichtzeichenanlagen für den Straßenverkehr (RiLSA)*. 2010.

[23] SCHLOTHAUER & WAUER. *LISA – Software-Paket zur Planung, Bewertung und Optimierung von LSA*. URL: https://www.schlothauer.de/software-lisa.

[24] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG, et al. “Gymnasium: A Standard Interface for Reinforcement Learning Environments”. In: *arXiv preprint arXiv:2407.17032* (2024).

[25] A. Müller and M. Sabatelli. “Safe and psychologically pleasant traffic signal control with reinforcement learning using action masking”. In: *25th IEEE ITSC*. 2022. URL: https://ieeexplore.ieee.org/abstract/document/9922306.

[26] FGSV Verlag GmbH. *Handbuch für die Bemessung von Straßenverkehrsanlagen. Technisches Regelwerk. Teil S - Stadtstraßen*. 2015.

A Model Learning Perspective on the Complexity of Cyber-Physical Systems

Nemanja Hranisavljevic[1], **Tom Westermann**[1], **Swantje Plambeck**[2], **Henrik Sebastian Steude**[1], **Gesa Benndorf**[3], **and Oliver Niggemann**[1]
[1]Institute of Automation Technology, Helmut Schmidt University, Hamburg, Germany
[2]Institute of Embedded Systems, Hamburg University of Technology, Hamburg, Germany
[3]Fraunhofer Center for Machine Learning, Fraunhofer IOSB-INA, Lemgo, Germany

Abstract

A large palette of models and their corresponding learning algorithms have been applied to time series observed from cyber-physical systems (CPSs). For some use cases, simple linear methods are sufficient, while for others, even sophisticated machine learning approaches fail to extract subtle patterns in system behavior. To date, the literature has not examined this phenomenon adequately and lacks a comprehensive analysis linking the characteristics of CPSs with the suitability of different models and learning algorithms.

In this work, after examining the complexity of multiple real-world and artificial CPS use cases, we identify several key aspects that distinguish them: 1) the number of system variables, 2) the degree of interdependence between discrete-event part and continuous part of the system, and 3) the number of unobserved system inputs. By analyzing the approaches successfully applied in the respective use cases, we were able to distill preferred techniques for addressing systems of different complexity levels.

Keywords Cyber-Physical Systems · Machine Learning · Model Learning · System Complexity · Data Dimensionality · Hybrid Dynamical Systems · Hybrid Automata

1 Introduction

Cyber-physical systems (CPSs) [19, 26] provide access to sensor measurements (such as speed or power consumption) and other variables (for example, set-point values in a control program) within a large number of devices. Together, these variables carry information about the holistic dynamic behavior of the system. This opens the door to various services based on artificial intelligence (AI), such as self-diagnosis [7] or self-reconfiguration [2], which typically rely on mathematical models of the dynamic behavior of the system. Machine learning (ML) can be used to create such models from historical data, such as system communications or sensor readings, even without much expert knowledge of the system.

The model learning and analysis of the system behavior during an actual operation phase is crucial for typical CPSs. However, at design time not all the information about the later operation is available or it is costly to gather. In such situations, the goal of model learning is to create a model of the dynamics of the system using the previously observed system data that represent training data in the ML methodology [4]. Then, given a series of successive system observations, the learned models can infer the probability of observing the given data, or predict future observations.

It is clear that many various models and approaches have been proposed to this day, from simple ones (such as nearest-neighbor-based [9]) to complex and highly parameterized ones (such as deep transformer neural networks [6]). However, the literature is not sufficiently transparent on the key aspects of the use cases studied, that could help us assess the suitability and performance of other model-learning approaches.

To address this gap, we propose the following research questions (RQs).

***(RQ 1)* From an ML perspective, how do the CPS model learning use cases differ from each other? What are simple and what are complex problems in this context?**

After investigating eight use cases from the literature, we adopt the well-established hybrid automata formalism [1, 20] as suitable for representing various CPSs. Using this representation, we propose a faceted classification scheme based on three crucial complexity aspects for the practitioner of ML (see Figure 1).[1]

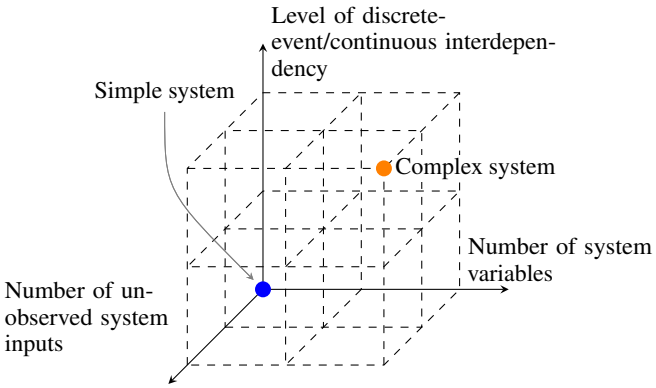


Figure 1: The three key dimensions of CPS system complexity.

***(RQ 2)* What state-of-the-art approaches are suitable for simple and what for more complex problems with respect to the three dimensions?**

We recognize ten relevant approaches from the literature that have been used effectively in anomaly detection, condition monitoring, and supervisory control. These approaches are applied to datasets of diverse degrees of complexity, according to the previously defined classification scheme. This allowed us to connect the characteristics of the methods used with the characteristics of the modeled system, as well as to explain the additional requirements and efforts required by the complex systems.

There is already a broad literature investigating the specifics of ML approaches applied in different application domains as well as in the CPS domain. For example, [15] provides an overview of model learning in industrial anomaly detection use cases. Similarly, [4] discusses the very general trends and requirements for ML in CPSs. Moreover, [18] evaluates residual-based anomaly detection approaches (autoencoder and input-output regression) on artificial CPS data. [12] compares several approaches in a manner similar to our work; however, it does not relate the facets of the system to the performance of different models. More recently, there has been work on creating scalable simulation models of different complexity (e.g. [8, 31]). In contrast, our work investigates the influence of multiple system facets on the choice of model learning techniques primarily in real-world CPS use cases.

The structure of the paper is the following. In Section 2, three aspects of system complexity are defined. Section 3 describes the selected state-of-the-art approaches. Section 4 discusses the reasons for (non-)suitability of these approaches to systems of different complexity. The conclusion and future work are given in Section 5.

[1] We limit ourselves to the three easily assessable dimensions, while other possibly relevant aspects, such as the order of system dynamics, application of the model or implemented control technique are not considered in this work.

2 CPS complexity dimensions

In order to answer *(RQ 1)*, we investigate multiple real-world and artificial use cases from the state-of-the-art literature. The use cases are selected to represent various CPS systems from different domains, including manufacturing, energy production, and space, as well as different tasks such as diagnosis or control.

1. Wind Power Plant (WPP) [10]
2. High-Rack Storage System (HRSS) [16]
3. Secure Water Treatment (SWaT) [13]
4. eBZ Assembly Plant (eBZ) [17]
5. Environmental Control and Life Support System of ISS (ECLSS) [23, 27]
6. PWM DC-DC buck converter (BuckConv) [3]
7. Excitable Cells (EC) [14]
8. Three Tank System (Tanks) [5]

In order to represent a CPS system, it is common to use the hybrid automata framework [1, 20], which is also adopted in this work (see Figure 2). The evolution of the continuous part is given by differential equations (*Diff. Eq.*), while the discrete-event dynamics is given by a transition model defining 1) switching of discrete system modes and 2) possible abrupt jumps in continuous variables. Switches are often associated with events (e_1 and e_2 in the figure), which can be triggered externally or follow from the continuous variables (e.g. when some condition turns satisfied). In addition, there is one externally triggered input e_1, and one continuous input $u_1(t)$ that affects the mode q_3.

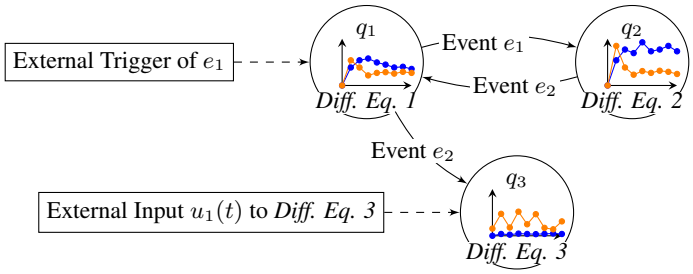


Figure 2: An illustration of a hybrid automaton with three modes (discrete states), three possible transitions and two continuous variables whose behavior is represented by different *Diff. Eq.*

In this work, we use the hybrid automata formalism to define the complexity aspects of CPSs. Namely, we represent CPS systems using a set of concurrent automata that can access each other's variables, trigger events, and share the same continuous part of the system. This is consistent with the distributed principles of CPS design (more information can be found in [1], as such details are outside the scope of this work).

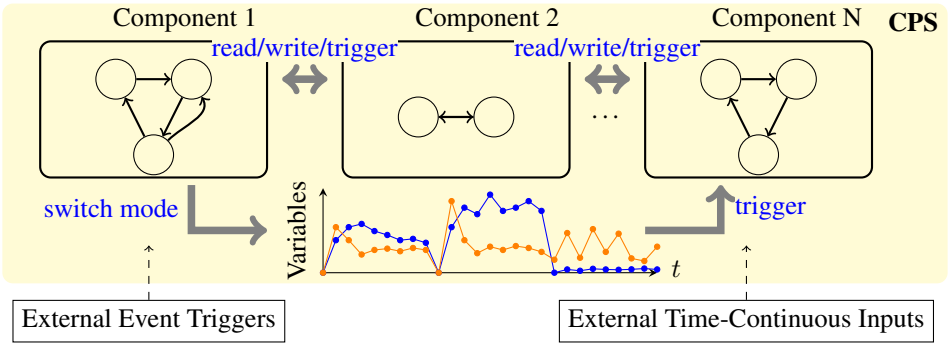


Figure 3: Set of concurrent automata as a representation of CPSs, used e.g. in [17].

2.1 Complexity Dimension 1: Number of observed system variables

It is well known that dealing with more variables of an unknown dependency structure implies a more difficult ML task, which might further lead to the so-called "curse of dimensionality" [11]. For this reason, we define the first aspect of system complexity as the number of system variables. This can be further decoupled into the number of discrete or categorical variables in the discrete event part of the system and the number of continuous variables. In the considered systems, anywhere between a few and hundreds of signals are analyzed jointly, which is undoubtedly an important parameter when choosing the model to learn. We categorize systems into three classes: less than 10 variables (class 1), 10 to 100 (class 2), and more than 100 variables (class 3) – from the simplest to the most complex, based on the information available in the literature (see Table 1). This classification takes into account solely the quantity of variables, disregarding the information they encompass, as assessing this information directly could be challenging.

System	Number of observed system variables	Class
WPP	12 continuous variables	2
HRSS	18 continuous and 3 discrete variables	2
SWaT	25 continuous and 26 discrete variables	2
eBZ	Hundreds of discrete signals published by several PLCs and the manufacturing execution system (MES)	3
ECLSS	Hundreds of continuous and discrete variables	3
BuckConv	6 continuous variables, 4 discrete variables	1
EC	7 continuous variables, 6 discrete variables	2
Tanks	10 continuous variables, 3 discrete variables	2

Table 1: Number of system variables in eight considered systems.

2.2 Complexity Dimension 2: Degree of discrete-event/continuous interdependency

In CPSs, model learning often does not consider the interactions between the discrete-event and continuous parts; then, the two parts are learned independently. Other approaches use the observed discrete data to split the continuous observations into segments corresponding to the same discrete state. However, changes in discrete signals do not always initiate mode changes in continuous behavior. This can be dealt with in different ways. For example, segments can be clustered (grouped) into those with similar behavior, or modes can be identified solely from continuous variables. Finally, the influence of continuous variables on discrete behavior is achieved by triggering a discrete state transition when conditions are satisfied. The variety of existent techniques to capture interdependency between discrete-event and continuous parts is the reason why we propose the degree of interdependency as a second complexity dimension.

It is not straightforward to evaluate the interdependency of discrete-event and continuous variables. One possible solution would be to calculate the Pearson correlation coefficient between the sequence of discrete variables and the sequence of continuous variables. Another possible solution could be to count the number of discrete (continuous) variables that are involved in the continuous (discrete) execution. In this work, we give only a rough estimate of the complexity level (1,2,3) based on the description of the system in the respective literature (see Table 2), such that:

1. if the discrete-event and continuous parts are modeled separately, we assign level 1 to the system,
2. if the influence is modeled only in one-way, we designate it as level 2, and,
3. if it was necessary to model it bidirectionally, then we assign level 3 to the system.

System	Level of discrete-event/continuous interdependency	Class
WPP	Only continuous variables are modeled in order to estimate normal functioning of the system. Clusters in the continuous behavior are only implicitly modeled.	1
HRSS	The position sensors trigger the changes of discrete modes in conveyors while each of the conveyors can be on or off causing different behavior of power, voltage and other signals.	2
SWaT	Many discrete and continuous variables influencing each other.	3
eBZ	The system has mostly discrete variables working in a timed manner and based on the temporal dependency of discrete variables.	1
ECLSS	Many discrete and continuous variables influencing each other.	3
BuckConv	Discrete events triggering state transitions depend on continuous variables	2
EC	Discrete variables are thresholded output variables.	2
Tanks	Discrete variables only enable mode. transitions	1

Table 2: Level of discrete-event/continuous interdependency in eight considered systems.

2.3 Complexity Dimension 3: Number of unobserved input variables

The systems under consideration encompass multiple, and frequently numerous, signals. Typically, the observed data variables comprise signals pertinent to system behavior and are relatively straightforward to observe, given the availability of sensors and data acquisition capabilities. However, not all relevant input signals can be observed, which can contribute to the stochastic behavior of the system, behavior that remains unexplained by the observed variables. Modeling systems characterized by significant stochastic behavior presents a more complex learning task. The categorization of the systems discussed is based on the estimated number of unobserved inputs (disturbances) that impact the system dynamics. They are classified into three distinct categories: those free from unobserved inputs (class 1), those containing fewer than five (class 2), and those comprising more than five unobserved inputs (class 3).

System	Number of unobserved system variables	Class
WPP	Discrete variable representing current gear of the system and continuous power consumption are some of the unobserved signals.	2
HRSS	Relevant discrete signals are available (HRSS v1), but not used in some experiments (HRSS v2).	3
SWaT	All relevant signals are considered to be available.	1
eBZ	All relevant signals are considered to be available.	1
ECLSS	Besides user inputs such as set-point adjustments, the system is influenced by short-term and long-term trends driven by the space station's orbit, affecting operational dynamics. Presence and other behavior of astronauts is also unobserved.	3
BuckConv	All relevant signals are available.	1
EC	Only voltages are observed. Internal currents and state variables are not observed.	2
Tanks	Only one tank level is observed while other tank levels are hidden.	2

Table 3: Number of unobserved system variables in eight considered systems.

System classification in 3 dimensions

In order to visualize the complexity of the use cases from the literature we create a 3D scatter plot (see Figure 4). Several points appear obvious:

- As expected, synthetic datasets are typically simpler than real-world ones.
- To some extent, complexity tends to grow along all dimensions simultaneously.
- There is significant diversity among the systems considered.

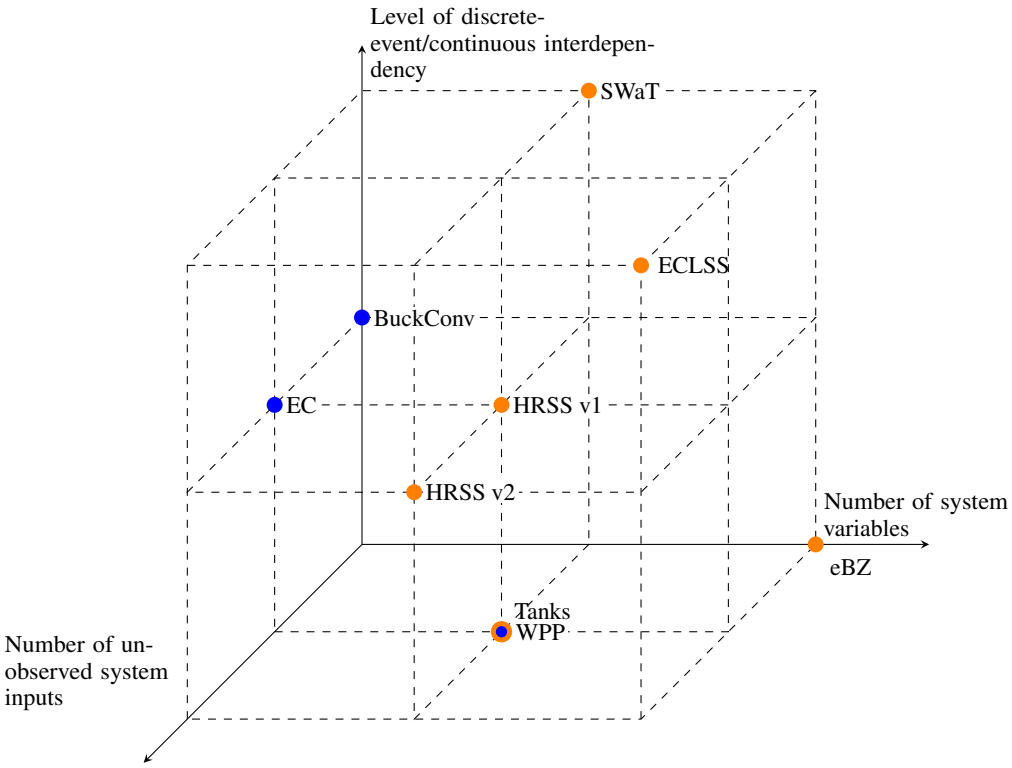


Figure 4: The three key dimensions of CPS system complexity. Blue circles represent synthetic and orange circles represent real-world use cases.

The levels of complexity dimensions are selected exactly to highlight the differences between the systems, while the complexity values of particular systems are typically not too difficult to estimate. In this way, with the proposed faceted scheme, we answer *(RQ 1)* of this work.

3 Model learning approaches

This section presents a range of model-learning approaches that have been proposed in the literature to model the respective systems in Section 2. Each approach is briefly described in terms of its methodological framework, the underlying assumptions, and specific implementation details. This overview sets the stage for a subsequent evaluation (Section 4) of their suitability across varying complexity profiles. The approaches examined, from oldest to youngest, are as follows.

- 2012 [22] In the HyBUTLA approach, switches of continuous dynamics appear as: 1) controlled switches as a consequence of a change (discrete event) in the discrete data. The continuous data are then segmented whenever a change in the discrete data appears. 2)

autonomous switches, which are not manifested in the discrete data and must be identified in order to segment the continuous data into segments that belong to the same mode (ODE). This is achieved using wavelet analysis to detect switches in the continuous behavior and by clustering the model parameters for each segment. The behavior of the discrete part is learned by comparing sequences of future events for pairs of segments in a specific bottom-up order.

- 2014 [21]: OTALA (Online Timed Automaton Learning Algorithm) is an online, passive algorithm for automatically identifying timed behavior models in production systems, designed specifically for discrete data. It learns from real-time observed data without needing stored datasets or system queries. OTALA maps unique signal vectors to system states and captures timed events to model dynamic behaviors. The resulting timed automaton supports real-time diagnostics by detecting anomalies and deviations efficiently.
- 2015 [9]: A static approach is presented which combines PCA (used to reduce data dimensionality) and the nearest-neighbor method to estimate the regions of normal data points. While this method focuses on the stochastic nature of the system, it aims not in modeling system dynamics. The proposed approach is compared with DBSCAN and spectral clustering which were outperformed.
- 2019 [10]: AE A static deep autoencoder architecture is applied to capture the stochastic nature of observed data points. Similarly to some previous approaches, it lacks extension to capture dynamics.
- 2020 [16]: DENTA (DEep Network Timed Automaton) approach is based on the following: 1) The DENTA network: a stacked network of restricted Boltzmann machines to extract a latent binary representation of time series data; 2) The DENTA automaton: a transition model of the latent binary variables which is learned in the second step from the encoded binary representation. The latent automaton model incorporates the probability distribution of the next occurring event given the current state of the automaton.
- 2021 [25]: POSEHEAD is an offline learning strategy for hybrid automata. First, a sliding-window strategy is applied to segment signals. Then, similar segments are detected using dynamic time warping. Transitions are derived on the basis of changes in the inputs as well as the residence time in a state. Finally, continuous flow functions of the dynamical modes are identified with polynomial regression.
- 2022 [30]: HAutLearn is a learning algorithm for hybrid automata with four major steps. First, change points are identified in the signals. Afterwards, segments with similar solution spaces are identified using LMI. In the third step, event conditions in the form of linear inequalities are found using the random sample consensus. Finally, the modes are merged and folded into a hybrid automaton model from a prefix tree acceptor.
- 2022 [29]: In this work, a model of a dynamical system is learned as a heterogeneous Petri net using the DyClee clustering algorithm (Dynamic Clustering algorithm for tracking Evolving Environments). With DyClee, multidimensional data from a system is clustered adaptively over time. The clustering consists of a distance-based clustering in a first stage and a density-based clustering in a second stage. The density-based stage again consists of a global and a local clustering. To avoid a functionality shift over time, a forgetting process in the form of a decay function is included.
- 2023 [28]: In [28], a TCN-VAE based seq2seq model is introduced that learns discrete and continuous variables simultaneously. The model is designed to capture all unobservable variables and hidden system states during training, encoding them in the latent space of the multivariate time series data it models. This architecture employs a specialized latent space, explicitly designed to enable anomaly detection at the subsystem level. The so-called composite latent space is structured to reflect the subsystem layout of a CPS, facilitating both failure isolation and the identification of cross-subsystem anomalies.
- 2024 [24]: FaMoS learns the dynamic model of a hybrid system in four steps: trace segmentation, segment clustering, mode identification, and model construction. The segmentation and clustering are based on the similarity of signals, i.e., Euclidean distance and dynamic time warping. The continuous dynamics in the mode identification phase is characterized by differential equations. Finally, the model is represented in a decision tree, which evaluates to the currently active mode based on the previous mode and the input.

4 Interweaving system and model characteristics

In this section, we evaluate the approaches described previously with respect to the complexity of the use cases in which they were applied. Rather than evaluating the methods in isolation, the analysis focuses on how each approach performed in specific contexts, characterized by the degrees of complexity introduced in Section 2. This evaluation provides insight into the suitability of different methods based on the complexity profiles of their real-world applications, highlighting trends in their effectiveness and adaptability – which is *(RQ 2)* of this work.

Figure 5 shows the complexity levels of the system(s) to which each of the investigated approaches has been successfully applied. The following conclusions can be inferred:

- Neural network-based approaches (DENTA, Composite-Latent-TCN-VAE) are more common for more complex systems, especially for the third dimension of complexity (unobserved system inputs). The affected systems have complexity (2,2,3) and (3,3,3), where used triplets represent three dimensions of complexity.
- Interestingly, when only continuous behavior is modeled in systems with complexity (·,1,·), static approaches (neglecting sequentiality of the data) are often sufficient (AE, PCA + kNN).
- The proposed approaches based on linear systems (HAutLearn, FaMoS, POSEHEAD) are more common for simple and artificial datasets of complexity (1,2,1) or (1,2,2).
- The continuous-only use cases of complexity (2,1,2) are common and addressed by diverse techniques (AE, PCA+kNN, DyClee).

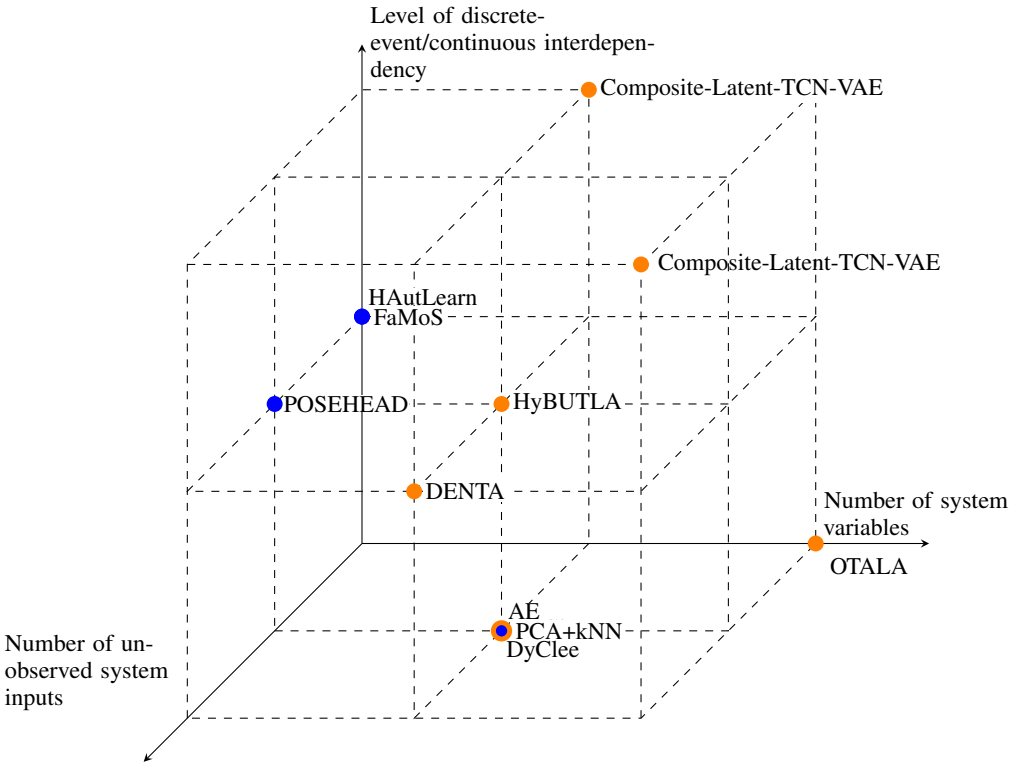


Figure 5: The state-of-the-art approaches visualized on the faceted classification scheme. Blue circles represent synthetic and orange circles represent real-world use cases.

5 Conclusion and future work

Various solutions for model learning have been proposed in the literature. However, it is hard to answer the question of their compliance with other CPS use cases. This work contributes in this direction by 1) proposing three aspects of system complexity which are relevant for model learning in CPS data; 2) analyzing methods based on their suitability to the systems of different complexity.

By interweaving system and model characteristics, this work evaluates the examined approaches with respect to the complexity of the use cases in which they were applied. Rather than evaluating the methods in isolation, the analysis focuses on how well each approach performed in specific contexts characterized by degrees of complexity. This evaluation provides insights into the suitability of different methods based on the complexity profiles of their applications.

Consequently, for a novel use case, upon categorizing a system according to its complexity, a machine learning practitioner is able to ascertain the relevance of various model-learning techniques.

Future research efforts should focus on incorporating additional use cases and exploring new methodological approaches. Furthermore, evaluating the methodologies on the same datasets would substantiate the validity of the proposed classification framework.

Acknowledgements

This research as part of the projects (K)ISS and ProMoDi is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr which we gratefully acknowledge. dtec.bw is funded by the European Union NextGenerationEU. This work was also partially developed within the Fraunhofer Cluster of Excellence "Cognitive Internet Technologies".

References

[1] R. Alur. Principles of Cyber-Physical Systems. Technical report, 2015.

[2] K. Balzereit and O. Niggemann. AutoConf: A New Algorithm for Reconfiguration of Cyber-Physical Production Systems. *IEEE Transactions on Industrial Informatics*, 2022. Publisher: IEEE Computer Society.

[3] O. A. Beg, H. Abbas, T. T. Johnson, and A. Davoudi. Model validation of pwm dc–dc converters. *IEEE Transactions on Industrial Electronics*, 64(9):7049–7059, 2017.

[4] J. Beyerer and O. Niggemann. Machine Learning in Automation. *At-Automatisierungstechnik*, 66(4):281–282, 2018.

[5] B. O. Bouamama, R. M. Alaoui, P. Taillibert, and M. Staroswiecki. Diagnosis of a two-tank system. Technical report, Intern Report of CHEM-project, USTL, 2001.

[6] Z. Chen, D. Chen, X. Zhang, Z. Yuan, and X. Cheng. Learning Graph Structures With Transformer for Multivariate Time-Series Anomaly Detection in IoT. *IEEE Internet of Things Journal*, 9(12):9179–9189, 2022.

[7] A. Diedrich, A. Maier, and O. Niggemann. Model-based Diagnosis of Hybrid Systems using Satisfiability Modulo Theory. Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19), 2019.

[8] J. Ehrhardt, M. Ramonat, R. Heesch, K. Balzereit, A. Diedrich, and O. Niggemann. An AI benchmark for Diagnosis, Reconfiguration & Planning. Sept. 2022.

[9] J. Eickmeyer, P. Li, O. Givehchi, and O. Niggemann. Data Driven Modeling for System-Level Condition Monitoring on Wind Power Plants. In *In: The 26th International Workshop on Principles of Diagnosis (DX-2015)*, 2015.

[10] B. Eiteneuer, N. Hranisavljevic, and O. Niggemann. Dimensionality Reduction and Anomaly Detection for CPPS Data using Autoencoder. In *20th IEEE International Conference on Industrial Technology*, volume 2019-Febru, Melbourn, Australia, Mar. 2019.

[11] J. H. Friedman. On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. *Data Min. Knowl. Discov.*, 1(1):55–77, 1997. Publisher: Kluwer Academic Publishers Place: Hingham, MA, USA.

[12] A. Garg, W. Zhang, J. Samaran, R. Savitha, and C.-S. Foo. An Evaluation of Anomaly Detection and Diagnosis in Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–10, Aug. 2021.

[13] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, editors, *Critical Information Infrastructures Security*, pages 88–99, Cham, 2017. Springer International Publishing.

[14] R. Grosu, S. Mitra, P. Ye, E. Entcheva, I. V. Ramakrishnan, and S. A. Smolka. Learning cycle-linear hybrid automata for excitable cells. In A. Bemporad, A. Bicchi, and G. Buttazzo, editors, *Hybrid Systems: Computation and Control*, pages 245–258, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[15] A. Happonen, U. Usmani, and J. Watada. A Review of Unsupervised Machine Learning Frameworks for Anomaly Detection in Industrial Applications. pages 158–189. July 2022.

[16] N. Hranisavljevic, A. Maier, and O. Niggemann. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Engineering Applications of Artificial Intelligence*, 95:103826, 2020.

[17] N. Hranisavljevic, T. Westermann, P. Kroke, and C. Waschkies. Using ML-Based Models in Simulation of CPPSs: A Case Study of Smart Meter Production. In O. Niggemann, J. Beyerer, M. Krantz, and C. Kühnert, editors, *Machine Learning for Cyber-Physical Systems*, pages 19–29, Cham, 2024. Springer Nature Switzerland.

[18] C.-C. Hsu, G. Frusque, and O. Fink. *A Comparison of Residual-based Methods on Fault Detection*. Sept. 2023.

[19] E. A. Lee. Cyber Physical Systems: Design Challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369, 2008.

[20] J. Lunze and F. Lamnabhi-Lagarrigue. *Handbook of Hybrid Systems Control: Theory, Tools, Applications*. Cambridge University Press, 2009.

[21] A. Maier. Online Passive Learning of Timed Automata for Cyber-Physical Production Systems. In *The 12th IEEE International Conference on Industrial Informatics (INDIN 2014)*. Porto Alegre, Brazil, July 2014.

[22] O. Niggemann, B. Stein, A. Vodenčarević, A. Maier, and H. Kleine Büning. Learning Behavior Models for Hybrid Timed Systems. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1083–1090, Toronto, Ontario, Canada, 2012.

[23] P. Parodi, Z. Szigetvari, R. Muller, and A. Quaglia. Columbus eclss: five years of operations and lessons learned. In *43rd International Conference on Environmental Systems*, page 3414, 2013.

[24] S. Plambeck, A. Bracht, N. Hranisavljevic, and G. Fey. FaMoS– Fast Model Learning for Hybrid Cyber-Physical Systems using Decision Trees. In *Proceedings of the 27th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '24, New York, NY, USA, 2024. Association for Computing Machinery. event-place: Hong Kong SAR, China.

[25] I. Saberi, F. Faghih, and F. S. Bavil. A Passive Online Technique for Learning Hybrid Automata from Input/Output Traces. *ACM Transactions on Embedded Computing Systems*, 22(1):1–24, 2021.

[26] J. Schlick. Cyber-physical systems in factory automation - Towards the 4th industrial revolution. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, page 55, May 2012. ISSN: Pending.

[27] H. S. Steude, C. Geier, L. Moddemann, M. Creutzenberg, J. Pfeifer, S. Turk, and O. Niggemann. End-to-end MLOps integration: a case study with ISS telemetry data. In *ML4CPS – Machine Learning for Cyber-Physical Systems*. Helmut-Schmidt-Universität Hamburg, 2024.

[28] H. S. Steude, L. Moddemann, A. Diedrich, J. Ehrhardt, and O. Niggemann. Diagnosis driven Anomaly Detection for CPS. *arXiv preprint arXiv:2311.15924*, 2023.

[29] A. Vignolles, E. Chanthery, and P. Ribot. Hybrid Model Learning for System Health Monitoring. In *Symposium on Fault Detection, Supervision and Safety for Technical Processes (SAFEPROCESS)*, pages 7–14, Pafos, 2022. Science Direct.

[30] X. Yang, O. A. Beg, M. Kenigsberg, and T. T. Johnson. A Framework for Identification and Validation of Affine Hybrid Automata from Input-Output Traces. *ACM Transactions on Cyber-Physical Systems*, 6(2):1–24, 2022.

[31] B. Zimmering, O. Niggemann, C. Hasterok, E. Pfannstiel, D. Ramming, and J. Pfrommer. Generating Artificial Sensor Data for the Comparison of Unsupervised Machine Learning Methods. *Sensors*, 21:2397, 2021.

CHALLENGES AND OPPORTUNITIES IN DEVELOPING INN-BASED CONTROL SYSTEMS FOR MODULAR DRONES

Christian Wittke[1], **Robert Ludwigs**[3], **Mark Tappe**[1], **Markus Schatz**[2], **Achim Kampker**[3], **and Oliver Niggemann**[1]
[1]Computer Science in Mechanical Engineering
[2]Fluid Machinery for Energy Technology
[1,2]Helmut Schmidt University, Holstenhofweg 85, 22043 Hamburg
[3]Chair of Production Engineering of E-Mobility Components (PEM), Faculty of Mechanical Engineering, RWTH Aachen, 52062 Aachen

March 20, 2025

ABSTRACT

As drone technology evolves, modular drones are increasingly central, offering rapid adaptability through the interchange of sensors, motors, and structural battery modules. However, this flexibility also introduces complex control challenges that traditional Proportional-Integral-Derivative (PID) controllers often struggle to address, particularly under dynamic reconfigurations and nonlinear responses. In this paper, we propose a novel approach integrating Invertible Neural Networks (INNs) and Reinforcement Learning (RL) to enhance adaptability and effectiveness in modular drone control. INNs facilitate precise, reversible command mapping via bijective transformations, ensuring robust handling of changing drone weight, geometry, and functionality. When combined with RL, these networks further enable real-time optimization of flight performance, dynamically responding to shifts in operational conditions. We outline a comprehensive research agenda employing the PX4 simulation framework to benchmark INN- and RL-based methods against standard PID controllers, focusing on improved response times, reduced error rates, and better system resilience. The anticipated findings aim to substantiate the potential of these advanced control systems—particularly in conjunction with emerging structural battery designs—to significantly expand the capabilities and operational scope of next-generation unmanned aerial vehicle (UAVs) in real-world applications.

1 Introduction

The evolution of drone technology is increasingly being shaped by the integration of advanced control systems in unmanned aerial vehicles (UAVs) (1), especially modular drones. These drones represent a significant technological shift, catalyzed by the growing complexity and diversity of applications spanning agriculture, disaster recovery, and urban development (2; 3; 4; 5). Modular drones excel in their ability to swiftly adjust to specific operational demands by allowing for the easy interchangeability of key components including sensors, cameras, batteries, motors, and structural elements (6). This adaptability, however, introduces sophisticated

challenges in managing flight control systems—challenges that conventional Proportional-Integral-Derivative (PID) controllers are ill-equipped to handle (7).

Traditional PID controllers, while robust in static designs and predictable environments, struggle with the dynamic reconfigurations and nonlinear responses that are typical in modular drone systems (8). In response to these limitations, this paper proposes an innovative research agenda that incorporates invertible neural networks (INNs) and reinforcement learning (RL) to overhaul the control systems of modular drones. The ambition is to supplant PID controllers with a new system that not only elevates drone operability but also revolutionizes their adaptability to complex, evolving environments.

Central to this approach are INNs, which stand out for their ability to execute bijective mappings between inputs and outputs—each output precisely correlated to an input, ensuring reversibility. This characteristic is pivotal for the complex flight dynamics of modular drones, allowing each control action to be directly traced back to its corresponding state, thus providing enhanced control and diagnostic capabilities. Such features are indispensable as they facilitate the drone's seamless adaptation to changes in weight, shape, or functionality, eliminating the need for the extensive recalibration that traditional systems require.

Additionally, advances in modular battery systems significantly impact the adaptability of drones. High-energy-density Li-ion batteries, particularly in structural configurations, contribute to weight reduction and increased flight endurance, enabling drones to meet diverse energy demands while maintaining flexibility in their mission profiles.(9)

Moreover, the operational environments for modular drones are often marked by high degrees of complexity and unpredictability. Here, reinforcement learning becomes essential. Unlike traditional control methods that rely on responses to predefined scenarios, RL adapts and optimizes through realtime interactions with the environment, significantly enhancing the drones' decision-making capabilities (10). This allows drones to adeptly navigate and respond to new and changing conditions (11).

By merging reinforcement learning (RL) with invertible neural networks (INNs), our goal is to develop a dynamic control system that allows drone flight management to evolve continuously. This evolution is driven by the use of accumulated experiences to improve strategies and adapt to changes in the environment. Our research will investigate the integration of INNs and RL within modular drone technology, emphasizing how this method enhances both performance and flexibility, and opens up new possibilities for innovative research and applications in autonomous flight. This paper delves into the integration of INNs and RL in modular drone systems, underscoring the enhancement of drone adaptability and the opening of new avenues for research and applications in autonomous flying. By pushing the boundaries of current capabilities, this approach establishes a groundwork for future UAV design and operational innovations, providing a detailed insight into how INNs and RL can revolutionize drone technology and alter the landscape of UAV operations and development.

2 State of the Art

2.1 Modular Drone System

Modular drone systems represent a significant advancement in drone technology, offering customizable solutions(6) that can be adapted to various applications ranging from research to logistics and surveillance(12). Recent developments in this area have focused on enhancing the modularity of both hardware and software components to facilitate rapid configuration changes and maintenance, ensuring mission-specific optimizations(13). However, the control of these systems remains a challenge, particularly in managing the dynamic stability and payload distribution when modules are reconfigured or added(14).

2.2 PID Controllers

PID (Proportional, Integral, and Derivative) controllers have been foundational in drone control systems, valued for their simplicity and effectiveness under standard flight conditions. These

linear controllers provide stable flight dynamics and are extensively documented (15; 16), making them a staple in both academic and practical drone applications. However, they often falter when faced with non-linear dynamics and environmental uncertainties, which are common in more complex or unpredictable scenarios (17).

2.3 Invertible Neural Networks (INN)

Invertible Neural Networks (INNs) represent a significant advancement in control systems by providing bijective mappings that guarantee each input corresponds to a unique output. This exact reversibility enhances interpretability and accountability—qualities that are crucial in applications requiring precise control, such as image processing, density estimation, and dynamic system modeling.

2.3.1 Architectural Foundations

INNs employ specialized architectures ensuring that every transformation is reversible (18). A key component of these architectures is the *coupling layer*, which partitions the input vector into two subsets. A typical coupling layer can be mathematically described as:

$$y_1 = x_1, y_2 = x_2 \odot \exp(s(x_1)) + t(x_1),$$

where $x = (x_1, x_2)$ is the input vector, $y = (y_1, y_2)$ is the output vector, $s(\cdot)$ and $t(\cdot)$ are scaling and translation functions (typically modeled using neural networks), and $\odot$ denotes element-wise multiplication. This design simplifies the computation of the inverse since $x_1 = y_1$ is directly recovered and x_2 can be efficiently reconstructed via:

$$x_2 = (y_2 - t(x_1)) \odot \exp\big(-s(x_1)\big).$$

Notable architectures leveraging these principles include:

- **NICE:** The NICE model splits the input into two parts and applies an additive coupling transformation to one partition while leaving the other unchanged. This approach greatly simplifies the computation of both the forward and inverse mappings, as the unchanged portion acts as an anchor for the transformation.
- **Glow:** Building upon NICE, Glow introduces invertible 1x1 convolutions, which serve as learned permutation matrices. These convolutions enable the network to mix information across all dimensions of the input vector more effectively, thus capturing complex dependencies within the data. Additionally, Glow maintains tractable computation of the Jacobian determinant, which is essential for density estimation tasks.

In both architectures, the design of the coupling layers—whether through additive or affine transformations—and the inclusion of operations like invertible convolutions contribute to the network's ability to perform efficient, reversible computations. These architectural foundations not only facilitate precise control and data transformation but also serve as the building blocks for the further developments discussed later in this work.

2.3.2 Dimensionality Reduction Techniques

Advancements in dimensionality reduction through INNs focus on preserving data integrity while reducing computational complexity. Since invertible neural networks require matching input and output dimensions (19; 20), it is crucial to apply suitable dimensionality reduction for aligning sensor data (e.g., speed, orientation) with motor-control outputs.

Manifold learning assumes that, although data exists in a high-dimensional space, it resides along a lower-dimensional manifold. INNs are particularly adept at learning such manifold structures, mapping high-dimensional data to more compact representations without losing essential geometric and topological features. Models like Manifold Flow integrate manifold learning with flow-based methods, simplifying data structures and improving tasks such as density estimation and generative modeling (21).

Invertible Autoencoders. By merging traditional autoencoders with reversible transformations, invertible autoencoders achieve lossless compression and reconstruction, preserving a perfect bijection between input and latent spaces (22; 23). This ensures every original data point can be fully recovered, which is invaluable for feature extraction, unsupervised learning, and scenarios requiring efficient data transmission.

These dimensionality reduction approaches highlight the versatility of INNs for handling complex, high-dimensional drone data while retaining crucial structural information—a vital capability in modern big-data environments.

2.4 Reinforcement Learning for Navigation

2.4.1 Current RL Applications

Reinforcement learning has recently gained traction in autonomous navigation systems due to its ability to learn optimal actions through trial-and-error interactions with the environment. In drone navigation, RL methods have shown potential in complex scenario adaptation and pathfinding efficiencies (24), particularly in environments that traditional algorithms find difficult to navigate due to unpredictable obstacles and changing conditions(10).

2.5 Drone Battery Systems

The requirements for drone battery systems exhibit significant variation across diverse application domains. Exploration drones may demand extended range, while rescue drones prioritize high power output. Transport drones, conversely, necessitate substantial power capacity to accommodate heavy payloads. This diversity in requirements necessitates specific battery characteristics, including high volumetric energy density, which is crucial for prolonged flight durations; mechanical and chemical stability, which are paramount for safety-critical missions; power density, which is vital for applications with high current demands; and modular capacity expansion, which allows for flexible adaptation to varied mission profiles.(9)

Currently, Lithium Polymer (LiPo) batteries are prevalent in the drone industry due to their high power density and adaptable form factors. Conversely, Lithium-Ion (Li-ion) batteries offer higher energy density and more robust casings but are less flexible in design. Configuring multiple battery cells in series and parallel enables customization of capacity and voltage to meet motor requirements and optimize flight time. Innovations in structural battery systems are exploring the potential to reduce weight. Initial simulations indicate a 41% increase in hover time by integrating cylindrical Li-ion batteries into the drone's structure.(25) While rigid structural batteries have been established in the automotive sector, such as Tesla's honeycomb structure and BYD's Blade Battery, their mechanical integration into drones presents additional challenges.(26)

3 Solution

3.1 Modular Drone Setup

3.1.1 Core Quadcopter Design

Our modular drone system centers around a core quadcopter, which acts as the central unit housing essential navigational sensors and the main battery pack. This core is engineered to be robust yet lightweight, ensuring optimal balance and energy efficiency. It functions as the brain of the drone, controlling all flight operations and data processing. A sketch to the principal setup is shown in Fig.1 below.

3.1.2 Modular Integration

Connected to the core are two types of modular blocks designed for specific enhancements: *Motor Modules:* These include additional motors and propellers, seamlessly integrating into the drone's flight control system. This integration allows the core to automatically recognize and

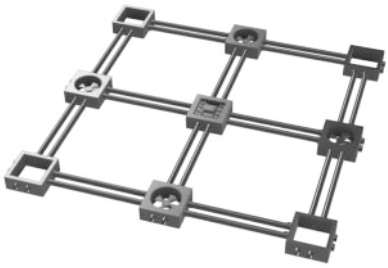


Figure 1: A first draft of our modular drone setup

synchronize the new motors, thus adapting flight dynamics to increased thrust.
Battery and Structural Modules: Supplementary battery units extend operational duration and contribute to the drone's structural integrity. These modules are designed to attach securely around the core, maintaining stability and balance even when configurations change. Moreover, the modular battery system allows for the dynamic selection of power sources based on mission-specific needs. High-energy-density Li-ion batteries provide extended endurance, while lightweight configurations enhance agility. Structural battery integrations further reduce the overall weight by merging energy storage with the drone's frame, thus optimizing both flight efficiency and mechanical durability.

3.1.3 Connectivity and Control

The interface between the core and the modular blocks features mechanical locks and electrical connectors that support quick and secure attachment. This modular connectivity allows for rapid reconfiguration in the field without tools. The drone's control system dynamically adjusts to new configurations by recalibrating flight controls in real time, based on inputs from the core's sensors and the current module setup. Development and Testing

Development involves iterative testing to refine module integration and evaluate the system under various conditions. Field tests will assess adaptability, endurance, and payload capacity, ensuring the drone meets diverse operational needs effectively.

3.2 Simulation Setup

The simulation setup designed to evaluate the performance of PID controllers, INNs, and RL includes the following key components:

3.2.1 Simulation Environment

The PX4 flight control software (27) offers a high-fidelity simulation environment that is renowned for its robust, open-source, and real-time flight capabilities. It supports a wide range of drone models and is designed to simulate realistic aerodynamics, sensor feedback, and environmental conditions. This platform facilitates the precise measurement of key performance indicators such as response times, error rates, and adaptability of each control method under realistic and challenging conditions. Additionally, PX4's extensive customization options allow researchers and developers to tailor the simulation parameters to fit specific scenarios or experimental needs. The software also integrates seamlessly with other simulation tools, enhancing its utility for complex, multisystem tests. The community-driven development model of PX4 ensures it stays at the forefront of technological advances, continuously improving its accuracy and reliability.

3.2.2 Task Scenarios

To rigorously test the capabilities of each control system, multiple flight scenarios are configured within the PX4 environment:
Navigation Task: Drones are required to navigate a predetermined path through a simulated urban environment with moving obstacles, testing the precision and responsiveness of the control systems.
Obstacle Avoidance: This scenario introduces random, dynamically appearing obstacles that require quick detection and evasion, challenging the control systems' response times and adaptability.
Environmental Adaptability: Drones face simulated wind gusts and other atmospheric disturbances within the PX4 environment, necessitating real-time adjustment of flight parameters to maintain stability and precise course adherence.

3.2.3 Data Collection

Metrics such as response time, error rate, adaptability, and computational overhead are automatically captured by the PX4 simulation software for each task, providing a robust data set for further detailed analysis. This data will help evaluate the comparative effectiveness and efficiency of the different control systems in various operational scenarios.

3.3 Research Questions

Building upon the foundation laid by our modular drone setup and comprehensive comparative framework, this subsection delves into the crucial research questions that will guide our investigation into the integration and optimization of control systems using Invertible Neural Networks (INNs).

RQ1: Can the integration of INNs into drone systems improve real-time decision-making and error correction capabilities under varying flight conditions?
RQ2: To what extent do trained invertible neural networks maintain strict invertibility in practical drone control applications, and how does any deviation from perfect invertibility affect the overall system performance?
RQ3: How do invertible neural networks and reinforcement learning compare in terms of training time and data efficiency when applied to drone control?
RQ4: What is the impact of model complexity on the accuracy and performance of drone navigation tasks when using INNs compared to RL?

4 Methodology

This section describes the methods used to evaluate the performance of traditional PID controllers and innovative control systems based on Invertible Neural Networks (INNs) using NICE or GLOW architectures. The experiments are conducted within the PX4 simulation framework, which offers a realistic environment for drone operations.

4.1 Experimental Design

The PID control system will be carefully calibrated with optimized gain settings to maintain stability and responsiveness under various simulated drone flight conditions. The calibration will cover essential functions such as altitude hold, position hold, and hoovering to a nearby position, reflecting basic operational scenarios for drones. These controllers will be integrated within the PX4 flight control stack, allowing them to interact directly with the drone's simulated sensors and actuators to implement control commands based on real-time feedback from the simulation environment.

Regarding the INN control system, the selection between NICE and GLOW architectures will depend on preliminary tests that assess each architecture's ability to handle dimensionality reduction and reversible computation, especially under limited computational resources. Data

preprocessing will be a critical initial step in model development, involving the normalization of high-dimensional sensory data from gyroscopes, accelerometers, and GPS modules. This normalization ensures data uniformity and enhances network training efficiency.

An advanced dimensionality reduction technique, such as invertible Autoencoders (iAE), will be used initially to pinpoint key features critical for drone control. This reduction is vital for maintaining computational efficiency and fitting the data within the constraints of onboard processing capabilities. The training process will employ a dataset generated from various simulated drone flights, capturing different weather conditions, payload weights, and flight dynamics to ensure that the model captures a comprehensive range of flight behaviors. After training, the INN models will be integrated into the PX4 simulation environment, setting up a communication pipeline where the INN outputs control commands directly to the drone actuators, interpreting and responding to real-time simulated environmental data.

4.2 Data Collection Protocol

The data collection protocol evaluates key performance metrics, including response time (latency between input sensing and motor output), error rate (trajectory deviations), adaptability (adjustment to changing conditions), and computational overhead (resource consumption during flight).

Testing begins in the PX4 simulation environment with PID controllers as a baseline for stabilization. Next, Invertible Neural Networks (INNs) are implemented to enhance stability by enabling precise hovering and automatic position correction under disturbances such as wind. After validating the INN, reinforcement learning (RL) is introduced to extend capabilities to complex navigation tasks, including obstacle avoidance and dynamic path adaptation. This structured approach ensures a stable foundation before advancing to more sophisticated control strategies.

To directly address our research questions:

RQ1: We will evaluate the INNs' capacity to enhance real-time decision-making and error correction during these stabilization and navigation phases, particularly under variable flight conditions introduced in our dynamic simulation scenarios.
RQ2: The strict invertibility of INNs in practical applications will be critically assessed, noting any deviations and their impacts on system performance during stabilization tasks and when responding to environmental perturbations.
RQ3: A comparative analysis of INNs and RL will be conducted focusing on their training time and data efficiency within the context of our simulation tests, providing insights into which technology proves more efficient in a controlled learning environment.
RQ4: Lastly, the impact of model complexity on the accuracy and performance of drone navigation tasks will be analyzed by comparing the RL navigation in INN-based systems with pure RL-based systems, aiming to identify any significant variances in operational efficiency due to the complexities of different models.

This structured testing and analysis will not only help us understand the capabilities and limitations of INNs and RL in drone control but also guide the development of more sophisticated algorithms tailored to enhancing drone operability in complex environments.

5 Results and Theoretical Considerations

This section outlines conceptual expectations rather than empirically validated findings. No practical or simulation-based experiments have been performed to date. All points below reflect theoretical assumptions guided by existing literature and preliminary feasibility assessments.

5.1 Advanced Control in Modular Drones

Integration of Reinforcement Learning We hypothesize that combining RL with an INN-based controller could enhance navigation efficiency, particularly by enabling drones to adapt in real time to new modules or payloads. In dynamic obstacle-rich environments, RL may facilitate smoother detours and faster learning curves than conventional PID approaches.

Expected Performance Gains Thanks to INNs' reversible mappings, flight parameters could be recalibrated on-the-fly as the drone's configuration changes (e.g., when adding extra motors or sensors). We anticipate improvements in both stability and responsiveness, but the **on-board computational load** remains an open question.

5.2 Battery Innovations and Endurance

Modular Battery Systems Introducing high-energy-density Li-ion modules that double as structural elements is projected to reduce overall weight and expand flight time. We expect these modular battery blocks to integrate seamlessly with the control logic, so that power management adapts in parallel with thrust and sensor configurations.

Potential Impact on Flight Efficiency If realized, structural battery packs could elevate both endurance and payload capacity. However, practical integration challenges (e.g., mechanical durability, thermal management) still require experimental validation before any definitive conclusions can be drawn.

5.3 Implications for Drone Technology

Adaptability and Robustness The anticipated synergy of RL, INNs, and modular battery systems may significantly broaden mission profiles for drones—ranging from time-sensitive deliveries to search-and-rescue scenarios. In principle, such a platform could handle frequent reconfiguration without extensive manual recalibration.

Next Steps Moving forward, we plan to validate these concepts through (1) PX4-based simulation studies encompassing different operational scenarios, and (2) limited real-world prototypes to verify endurance, computational feasibility, and safety under realistic conditions.

6 Future Research

Future investigations should further enhance the understanding and application of INNs, PID controllers, and RL in drone control systems. One promising direction is the exploration of advanced architectures and algorithms. Beyond NICE and Glow, models such as RealNVP (28) and FFJORD (29) may offer unique advantages in handling data flow and transformation. Additionally, research into hybrid control systems that combine PID controllers for basic stability with INNs for adaptive control and RL for strategic decision-making could lead to robust, efficient solutions for various CPS.

Enhanced learning techniques for RL represent another key area. Incorporating meta-learning strategies could significantly improve RL's ability to adapt to new environments or tasks, thereby reducing retraining time and data requirements (30). Similarly, transfer learning may enable RL models developed for one drone type or simulation environment to be adapted to other scenarios with minimal additional training.

Improving scalability and efficiency is also crucial. Optimizing INN models to reduce computational demands will make them more suitable for real-time applications on drones with limited processing power. Furthermore, enhancing the energy efficiency of these control systems is vital for prolonging drone operational time. Advancements in battery technology, such as integrating structural batteries into drone frames, could simultaneously provide energy storage and reduce weight, thereby boosting endurance.

Finally, real-world testing and validation are essential. While simulations offer invaluable initial insights, implementing these control systems in actual drone models and testing them in varied environmental conditions will be crucial for performance validation. This research should also address regulatory compliance and safety testing to ensure commercial and civil applicability. Moreover, tailoring these systems to specific applications—such as emergency response, traffic monitoring, or environmental surveying—and integrating user feedback will help enhance usability and overall effectiveness.

7 Conclusion

This paper has examined the theoretical potential of Invertible Neural Networks (INNs) for drone control in complex and dynamically changing environments, particularly with modular hardware configurations. While no direct experiments or simulations have been performed yet, existing literature and conceptual reasoning suggest that INNs could offer enhanced precision, adaptability, and real-time responsiveness compared to both traditional PID controllers and even certain reinforcement learning (RL) methods. As drone applications continue to expand into more demanding domains, advanced control paradigms like INNs will likely become increasingly important. This conceptual study underlines the need for further empirical validation—both in simulation-based frameworks and in real-world field tests—to confirm the scalability, robustness, and on-board feasibility of INN-driven control solutions. Ultimately, the insights gained here may help pave the way for future innovations in autonomous aerial vehicles and other cyber-physical systems.

References

[1] V. Kangunde, R. S. Jamisola, and E. K. Theophilus, "A review on drones controlled in real-time," *International Journal of Dynamics and Control*, vol. 9, no. 4, pp. 1832–1846, 2021.

[2] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani, "Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48 572–48 634, 2019.

[3] A. Gupta, T. Afrin, E. Scully, and N. Yodo, "Advances of UAVs toward Future Transportation: The State-of-the-Art, Challenges, and Opportunities," *Future Transportation*, vol. 1, no. 2, pp. 326–350, Sep. 2021.

[4] H. Liang, S.-C. Lee, W. Bae, J. Kim, and S. Seo, "Towards UAVs in Construction: Advancements, Challenges, and Future Directions for Monitoring and Inspection," *Drones*, vol. 7, no. 3, p. 202, Mar. 2023.

[5] E. C. Nunes, "Employing Drones in Agriculture: An Exploration of Various Drone Types and Key Advantages," Jul. 2023, arXiv:2307.04037 [cs].

[6] F. Schiano, P. M. Kornatowski, L. Cencetti, and D. Floreano, "Reconfigurable Drone System for Transportation of Parcels With Variable Mass and Size," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, 2022.

[7] J. Espinoza, N. Hakim, D. Tan, T. Wilson, K. Bingi, E. Khan, and S. Masrura, "Fractional-order pid control of quadrotor drone," in *2023 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2023, pp. 1–6.

[8] A. Zulu and S. John, "A Review of Control Algorithms for Autonomous Quadrotors," *Open Journal of Applied Sciences*, vol. 04, no. 14, pp. 547–556, 2014.

[9] D. S. Vohra, P. K. Garg, and S. K. Ghosh, "Power management of drones," in *Proceedings of UASG 2021: Wings 4 Sustainability*, ser. Lecture Notes in Civil Engineering, K. Jain, V. Mishra, and B. Pradhan, Eds. Cham: Springer International Publishing, 2023, vol. 304, pp. 555–569.

[10] A. A. Darwish and A. Nakhmani, "Drone navigation and target interception using deep reinforcement learning: A cascade reward approach," *IEEE Journal of Indoor and Seamless Positioning and Navigation*, vol. 1, pp. 130–140, 2023.

[11] I. Sharifi and A. Alasty, "Self-Tuning PID Control via a Hybrid Actor-Critic-Based Neural Structure for Quadcopter Control," Jul. 2023, arXiv:2307.01312 [cs, eess].

[12] B. Mu and P. Chirarattananon, "Universal flying objects: Modular multirotor system for flight of rigid objects," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 458–471, 2020.

[13] D. Hert, T. Baca, P. Petracek, V. Kratky, R. Penicka, V. Spurny, M. Petrlik, M. Vrba, D. Zaitlik, P. Stoudek, V. Walter, P. Stepan, J. Horyna, V. Pritzl, M. Sramek, A. Ahmad, G. Silano, D. B. Licea, P. Stibinger, T. Nascimento, and M. Saska, "MRS Drone: A Modular Platform for Real-World Deployment of Aerial Multi-Robot Systems," *Journal of Intelligent & Robotic Systems*, vol. 108, no. 4, 2023, arXiv:2306.07229 [cs].

[14] K. Moral, B. Ayran, and E. Altug, "Design and control of a modular multi-drone system with vertical assemble capability," *International Journal of Dynamics and Control*, Mar. 2024.

[15] H. Oersted and Y. Ma, "Review of pid controller applications for uavs," 2023.

[16] I. Lopez-Sanchez and J. Moreno-Valenzuela, "PID control of quadrotor UAVs: A survey," *Annual Reviews in Control*, 2023.

[17] J. Spencer, J. Lee, J. A. Paredes, A. Goel, and D. Bernstein, "An Adaptive PID Autotuner for Multicopters with Experimental Results," Sep. 2021, arXiv:2109.12797 [cs, eess].

[18] J. Kruse, L. Ardizzone, C. Rother, and U. Köthe, "Benchmarking invertible architectures on inverse problems," 2021.

[19] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," 2015.

[20] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," 2018.

[21] S. Li, H. Lin, Z. Zang, L. Wu, J. Xia, and S. Z. Li, "Invertible Manifold Learning for Dimension Reduction," Jun. 2021, arXiv:2010.04012 [cs].

[22] J.-H. Jacobsen, A. Smeulders, and E. Oyallon, "i-revnet: Deep invertible networks," 2018.

[23] B. Eiteneuer, N. Hranisavljevic, and O. Niggemann, "Dimensionality Reduction and Anomaly Detection for CPPS Data using Autoencoder," in *2019 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2019.

[24] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023.

[25] A. S. Hollinger, D. R. McAnallen, M. T. Brockett, S. C. DeLaney, J. Ma, and C. D. Rahn, "Cylindrical lithium–ion structural batteries for drones," *International Journal of Energy Research*, vol. 44, no. 1, pp. 560–566, 2020.

[26] X. Chen, Y. Xiang, J. Wu, F. Wu, S. Mei, X. Ye, H. Pan, Y. Xiang, X. Liu, F. Li, M. Huang, and X. Zhang, "Rigid structural battery: Progress and outlook," *Journal of Energy Storage*, vol. 91, p. 112070, 2024.

[27] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240.

[28] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real nvp," 2017.

[29] W. Grathwohl, R. T. Q. Chen, J. Bettencourt, I. Sutskever, and D. Duvenaud, "Ffjord: Free-form continuous dynamics for scalable reversible generative models," 2018.

[30] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning," *Science Robotics*, vol. 8, no. 82, p. eadg1462, 2023.

How to quantify the maturity of production processes

Negar Arabizadeh[1], **Julius Pfrommer**[2], **and Jürgen Beyerer**[1,2]
[1]Karlsruhe Institute of Technology, Vision and Fusion Laboratory (IES), Karlsruhe, Germany
[2]Fraunhofer IOSB, Karlsruhe, Germany

Abstract

Quantifying the properties of interest is a basic need in all fields of science and engineering. We propose formal definitions to quantify the maturity of production processes. The definitions are inspired by the concepts of controllability and observability from control theory. But instead of a binary classification of a system as controllable/uncontrollable and observable/unobservable, we consider a probability distribution for the remaining error. From the proposed maturity quantification, the remaining potential for optimizations can be evaluated. This is shown for the example of an electric arc furnace (EAF), for which high-fidelity simulation models are available.

Keywords Immature production processes · Maturity metrics for production processes · Complex system modeling

1 Introduction

A manufacturing process is immature if it cannot operate with the required productivity and product quality. New manufacturing processes involve a development phase before they are capable of production with the required quality, productivity, and cost-effectiveness. Abstractly speaking, a huge and complex optimization problem must be solved during the development phase in terms of the process structure, parameter values, control strategies, and process instrumentation (sensors and actuators).

Having metrics for measuring the maturity of a production process is essential, as it provides an overview of the process's development status in comparison with the desired goals of a mature process. Existing research on evaluating the maturity of production processes is mainly based on qualitative models that provide little or no quantitative meaning. On the other hand, quantitative models in the literature are primarily focused on measuring the Technology Readiness Level (TRL) rather than maturity [MMSM13]. Furthermore, there are no quantitative models that relate the internal features of the production processes (process variables and process dynamics) to the desired outcome of the production process, which is the quality of the final product. This gap motivated us to introduce new definitions to quantify the maturity of production processes, considering both the quality of the final product and the internal features of the process, and to investigate a relationship between them (such as a monotonic relation).

A qualitative metric in [Met11] presents a maturity assessment model to measure the maturity of social and technical systems. Also, due to the lack of knowledge on how to design theoretically sound and widely accepted maturity assessment models, the paper discusses the typical phases of maturity model development and application, taking a design science research perspective. In [Pfl95], a qualitative metric plan was proposed, which outlines five levels of process maturity and the corresponding types of metrics required for each level. In existing research, different

fields have developed quantitative measures for their areas of interest. In [BG16], a quantitative definition of risk for safety and security was developed. In [Bey99], a metric, Bayesian decision theory, was proposed to evaluate the relative worth of additional knowledge within the measurement context.

The scientific contribution of this paper is as follows:

- Definition of the maturity of a production process.
- Introducing quantitative definitions for measuring the maturity of production processes with the eventual goal of being able to relate the internal features of the dynamic system to the quality of the eventual product of the production process.
- Implementing our new definitions for quantifying maturity in an example of an electric arc furnace (EAF) model.

2 Related works

2.1 Maturity models in literature

In [PCCW93], the immature organization, especially in the context of software processes, is described as a reactionary organization that focuses on solving immediate crises, is not cost-efficient, is unscheduled, and, especially under tight deadlines, is unable to meet desired product quality and functionality. It has no objective way to judge the quality of the product or solve product or process problems, making the quality of production difficult to predict. As a result, activities aimed at improving quality are curtailed or eliminated. In literature, maturity is defined as "the state of being complete, perfect, or ready" [SS$^+$04]. For production processes, we can say that it shows evolutionary progress toward the target of the production process or the demonstration of a specific ability [Met11]. In [PCCW93], a "matured organization" is defined as an organization that follows the scheduled deadlines to prepare products and achieve the desired quality goals. It has management, less waste, the potential to improve, and is economically acceptable. Models for measuring the maturity of production processes on an ordinal scale are available in existing research. For example, [WK02] presents a maturity model, but it does not offer a sound mathematical definition for measuring maturity on a ratio scale. In [Met11] the concept of maturity assessment models for social systems was contextualized. In the literature, the terms "maturity" and "readiness" are sometimes used interchangeably, but there are some differences between them [SM96].

2.2 Generalizing controllability and observability

Classical definitions of controllability and observability in control theory

The original definition of controllability and observability from the control theory literature is as follows [Kal60]:

Controllability: The system state $\mathbf{s}$ is controllable if there exists an input signal $\mathbf{x}(t)$ over a finite time interval $0 \leq t \leq t_1$ such that a target state $\check{\mathbf{s}}$ is reached at t_1. When a state of the system is controllable, it is possible to control the system state to a desired point.

Observability: The system state $\mathbf{s}$ is observable if the state $\mathbf{s}(0)$ at time $t = 0$ can be fully determined by observations $\mathbf{y}(t)$ during a finite time interval $t_0 \leq t \leq 0$.

There are different methods in the literature, such as calculating the rank of the controllability matrix and the controllability Gramian matrix [Kal63], which indicate whether the system is controllable or not without providing any answer about the degree of controllability. Methods like [HN89, HS80, Tar92, VLL84] measure the degree of controllability of a system without any physical meaning. Therefore, measures based on minimum input energy are mostly used to give the degree of controllability a physical meaning.

Extending the classical definitions of controllability and observability

[MW72] proposed an infinite set of possible scalars to measure the degree of controllability and observability for linear dynamical systems. The physical interpretation of these measures is the minimum input energy required to regulate a system from its initial condition over a finite time interval. If the system requires less energy, it can be considered more controllable. [LP16] proposed a method to measure the degree of controllability of a system with unstable modes, based on the physical meaning of measuring the minimum energy required to change the state from an arbitrary initial condition to an arbitrary final condition within a positive time interval. Since the measure is related to the initial and final conditions, the appropriate initial and final conditions corresponding to the control objective should be used.

In an application, these measures were used to optimize certain structural parameters in order to improve the proposed quality measures. [MKG02, MKG04] proposed Gramian-based methods to determine the optimal locations of sensors and actuators. [SH05, SH06] extended these methods to nonlinear systems. In [KPPS09], the degree of controllability is defined as the minimum input energy required to achieve the final state zero in the presence of persistent external disturbance. It is composed of Gramian controllability and sensitivity matrices. In [ST24], VCS (Volumetric Controllability Score) and AECS (Average Energy Controllability Score) are defined. These are two mathematical definitions based on the Gramian controllability matrix and are unique solutions to the convex optimization problem, producing a degree for measuring the controllability of a system. In other words, the VCS of each state node indicates its importance in enlarging the controllability ellipsoid, and the AECS of each state node indicates its importance in steering overall states to a point on the unit sphere. This metric measures the energy input needed to change the state from one condition to the sphere. The characteristic function of the VCS is based on the log determinant of the controllability Gramian matrix, and the AECS is the trace of the inverse of this matrix. By the rule of duality, they could also provide insights into observability. [XYC$^+$18] introduced a measure for the degree of controllability of a linear system with random initial conditions and disturbance, by solving the problem of expected minimum energy transfer in fixed time and applying the method in turbines. In [SLM17], they developed a new data-driven method to measure the degree of controllability of a system using data.

The classical definitions of controllability and observability provide a definitive binary answer as to whether a system is controllable or observable. However, the classical definitions and their extensions in the existing research do not account for how much we are able to control or observe the system with a specific level of precision, considering the uncertainty. For example, in the case of controllability, the classical definitions do not account for how much we are able to control the system states to achieve their desired states with specific precision (similarly for observability). In the real world, uncertainty (such as output/input noise, process noise, etc.) prevents us from achieving the exact desired state with zero error. Instead, we can only reach the desired state within a certain level of precision. By considering uncertainty, we can evaluate the probability of successfully controlling the system states to the desired states with a specific degree of precision. A similar approach applies to observability, in terms of how much we are able to observe the states of the system with a specific level of precision.

By considering uncertainty and incorporating probability, we can better describe how effectively the system can be controlled or observed. Instead of simply stating whether the system is controllable or observable, we can assess how effectively we can bring the system states to their desired states or observe the system states with a specific level of precision. In this context, not only do we answer the questions of controllability and observability, but we also gain insights into the precision of these properties.

Consequently, an extension of these concepts is necessary, as we wish to capture the stochastic nature of production processes and the quality of their output. In ML, the concept of PAC learning (Probably Approximately Correct) [Val84] describes the number of data samples minimally required to achieve a model error smaller than some η with a probability larger than $1 - \delta$. We apply similar notions for generalizing the definitions of controllability and observability, considering the uncertainty that exists in the system and the precision we want to achieve to be able to quantify the maturity of the production processes.

3 Quantifying process maturity

As discussed in the previous section, there are different factors to consider when quantifying process maturity. For example, these include the ability to generate profit, the quality of the final product, the speed of the process, and the consumption of resources. Some aspects, such as the speed and quality of the process, depend on the product and can be used to quantify maturity, while others are related to the market conditions. The following is our new definition of a "mature production process":

Definition 1. *A production process is mature if it cannot be further improved economically.*

The economic argument is not really related to the physical production process itself, and we rather focus on measurable quantities for the physical production process and not for the market conditions, which are beyond the scope of this research.

As long as there is potential to improve the production process to meet the required threshold (considering its specific applications and economic aspects), it cannot be considered mature. This is not only due to technical considerations but also economic factors. Therefore, we integrate both aspects into our definition of maturity. In this context, we can determine the threshold at which the process is deemed sufficient, taking into account both its specific applications and economic feasibility.

For people in industry, the internal dynamics of the system is often not easily understood, comprehensible, or considered important or interesting. Their main concern is the economic aspect of the production process. In general, a mature system is one that prioritizes the economic efficiency of production.

The difference from previous definitions in the literature is that they focus on various factors, such as the efficiency of the production process, adherence to the schedule, and product quality. Since all of these factors ultimately impact the economic aspect of the production process, we can conclude that a system is mature when it is economically optimized. We introduce new definitions to quantify the maturity of production processes that relate the quality of the final product to its internal features.

3.1 Notation and definitions

The transformation of initially immature production processes into a higher maturity is achieved through a sequential approach of process development and subsequent expansion of production [FSL03]. Figure 1 shows the evolution of process development in which we present process instances after making software or hardware changes. If we make hardware changes, we move from the process instance P_i^j to the process instance P_{i+1}^j, and if we make parameter or software changes, we move from the process instance P_i^j to the process instance P_i^{j+1}. Changes can sometimes be done in parallel. After making these changes, there might be a possibility of achieving our desired mature process instance P^* or aborting, as shown by $\times$ in the figure. Figure 2 illustrates an idealized version of the closed-loop automated process instance P_i^j. It is made up of two main parts. One part shows the process variables and process dynamics, the input signal $\mathbf{x}$, noise $\mathbf{n}$ that might occur in the system, the state variables $\mathbf{s}$ of the dynamic system, the parameters of the system $\boldsymbol{\theta}$, the controller π and the output signal $\mathbf{y}$ which is a function of $\mathbf{x}, \mathbf{s}, \boldsymbol{\theta}$, and $\mathbf{n}$. The signals $\mathbf{x}$ and $\mathbf{y}$ are defined to be observable and $\mathbf{s}$ subsumes all unobservable dynamic quantities. If an added sensor or actuator instrumentation allows the components of $\mathbf{s}$ to be determined passively or actively, these quantities are assigned to be additional components of $\mathbf{x}$ and $\mathbf{y}$ in the next process instance P_{i+1}^j. The other part illustrates the input process material $\boldsymbol{\varepsilon}$ such as reactants and components with properties $\boldsymbol{\phi}$ and its flow to the product p with its properties $\mathbf{q}(p)$.

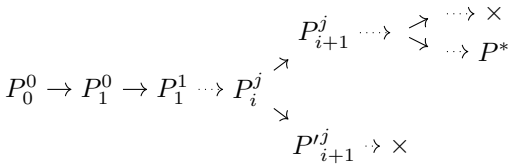


Figure 1: Evolution of the maturation of a production process. The symbol $\times$ indicates abortion, and P^* denotes the desired final mature process instance (subscripts denote the version with respect to structure, and superscripts enumerate soft improvements). If promising, it is conceivable to split the process maturation evolution into competing, parallel paths that pursue alternative process instances, here indicated as P'.

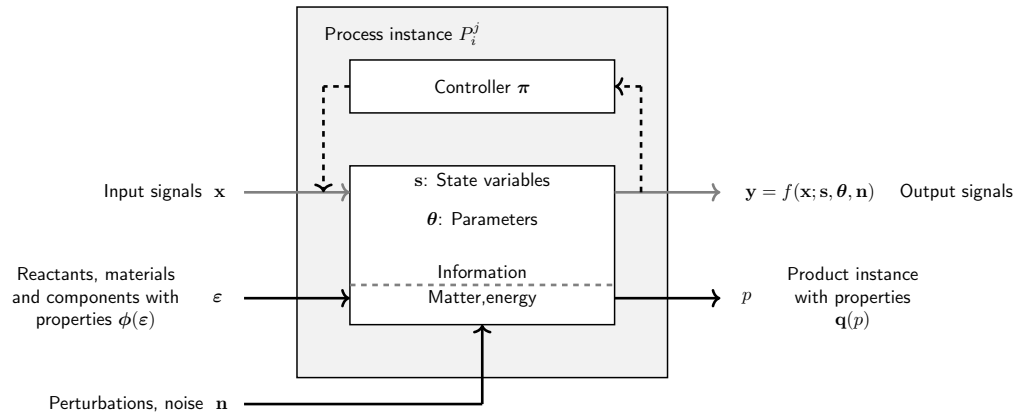


Figure 2: An abstract view of an idealized version of the closed-loop automated process instance. This may represent the complete production process or a sub-process at any hierarchical process level. The variables $\mathbf{x}$, $\mathbf{y}$, $\mathbf{n}$, and $\mathbf{s}$ are time-dependent signals. In fact, all involved variables should be marked with the indices ^j_i, since they refer to the process instance P_i^j, but this is suppressed whenever the context is clear.

3.2 New definitions to quantify process maturity

Elucidability

From here we consider the maturity of a process instance. For simplicity, we refer to it as P instead of P_i^j or do not even mention it, when the meaning is clear from context. For the process instance P_i^j, Elucidability which is a generalization of observability, is approached by defining the minimum observation time τ that is necessary to achieve a certain estimation precision for the state $\mathbf{s}$ and the parameters $\boldsymbol{\theta}$. We then find the probability of having the estimation error less than a constant precision $\eta_{\boldsymbol{\theta}}$ and $\eta_{\mathbf{s}}$. The internal states of the process can be estimated from the time series $\mathcal{X}_T$, $\mathcal{Y}_T$ during the finite observation duration T. Thus, $\mathcal{X}_T := \mathbf{x}(0), ..., \mathbf{x}(T)$ and $\mathcal{Y}_T := \mathbf{y}(0), ..., \mathbf{y}(T)$. The formula for Elucidability is:

$$E(T, \eta_s, \eta_{\boldsymbol{\theta}}) := \Pr(\|\mathbf{s}(T) - \hat{\mathbf{s}}(\mathcal{X}_T, \mathcal{Y}_T)\| < \eta_s \wedge \|\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}(\mathcal{X}_T, \mathcal{Y}_T)\| < \eta_{\boldsymbol{\theta}} | \mathbf{s}(0)) \quad (1)$$

Here $\hat{\mathbf{s}}(\mathcal{X}_T, \mathcal{Y}_T)$ is the estimation of the state $\mathbf{s}(T)$ at time T, with time series $\mathcal{X}_T$ and $\mathcal{Y}_T$ over duration T. The same applies to $\hat{\boldsymbol{\theta}}(\mathcal{X}_T, \mathcal{Y}_T)$, which is an estimation of $\boldsymbol{\theta}$. If the time duration T is not fixed for a given process instance, an appropriate *Dual Elucidability*, inspired from PAC learning [Val84], is defined as the minimum time to achieve a required error with a probability higher than $1 - \delta$.

$$\tau_E(\eta_{\mathbf{s}}, \eta_{\boldsymbol{\theta}}, \delta) := \min\{T > 0 | E(T, \eta_{\mathbf{s}}, \eta_{\theta}) > 1 - \delta\}. \quad (2)$$

Assuming that the Elucidability monotonically increases with the time T, by construction, E can be recovered from a characterization of τ_E and vice versa. Note that τ_E tends to infinity if a required estimation error cannot be achieved for any duration of the observations.

Forcability

Forcability F, which is the generalization of controllability, is defined to quantify the ability to steer the values of $\mathbf{y}$ and $\mathbf{s}$ via the control input $\mathbf{x}$ towards the target values $\breve{\mathbf{y}}$ and $\breve{\mathbf{s}}$. The most effective way to steer the process is to establish a closed-loop control system that reacts instantaneously to the dynamic responses of the process. To quantify the Forcability of a process instance with a given controller π, the probability of being able to force the process after time T into the neighborhood of target values $\breve{\mathbf{y}}$ and $\breve{\mathbf{s}}$ is defined as follows:

$$F_\pi(T, \eta_\mathbf{s}, \eta_\mathbf{y}) := \Pr(\|\mathbf{y}(T) - \breve{\mathbf{y}}\| < \eta_y \wedge \|\mathbf{s}(T) - \breve{\mathbf{s}}\| < \eta_\mathbf{s} \,|\, \mathbf{s}(0), \mathbf{x}(t) = \pi(\mathcal{Y}_t), \breve{\mathbf{y}}, \breve{\mathbf{s}}) \quad (3)$$

In closed loop control, $\mathcal{X}_T$ is determined by the control policy π. In the case where we do not have $\mathbf{y}(T)$ and $\mathbf{s}(T)$ and need to estimate them, we can replace $\mathbf{y}(T)$ and $\mathbf{s}(T)$ by the estimates $\hat{\mathbf{y}}(T)$ and $\hat{\mathbf{s}}(T)$. Thus, the modified formula for an estimation of Forcability is:

$$\hat{F}_\pi(T, \eta_\mathbf{s}, \eta_\mathbf{y}) := \Pr(\|\hat{\mathbf{y}}(T) - \breve{\mathbf{y}}\| < \eta_\mathbf{y} \wedge \|\hat{\mathbf{s}}(T) - \breve{\mathbf{s}}\| < \eta_\mathbf{s} \,|\, \mathbf{s}(0), \mathbf{x}(t) = \pi(\mathcal{Y}_t), \breve{\mathbf{y}}, \breve{\mathbf{s}}) \quad (4)$$

To achieve a final statement about Forcability, the optimal control strategy $\mathbf{x}(t) = \pi^*$ has to be applied.

$$F(T, \eta_\mathbf{s}, \eta_\mathbf{y}) := F_{\pi^*}(T, \eta_\mathbf{s}, \eta_\mathbf{y}) \quad (5)$$

With that, analogous to Elucidability, Forcability is defined as the probability of achieving target values $\breve{\mathbf{y}}$ and $\breve{\mathbf{s}}$ with deviation smaller than $\eta_\mathbf{y}$ and $\eta_\mathbf{s}$ applying the optimal control strategy π^*. Also in analogy to Elucidability, the *Dual* of Forcability is the minimum time required to achieve the desired remaining error under the regime of the optimal controller π^* with a sufficiently high probability $1 - \delta$.

$$\tau_F(\eta_\mathbf{s}, \eta_\mathbf{y}, \delta) = \min\{T > 0 | F(T, \eta_\mathbf{s}, \eta_\mathbf{y}) > 1 - \delta\}. \quad (6)$$

Supervisability

Since the product instances p delivered by the process and their quality are the ultimate objectives of production, it seems reasonable to base a maturity measure on the quality of the product. Supervisability S is defined to measure the maturity of the process instance P_i^j. Based on the probability of $\mathbf{q}(p)$ falling in the desired set Q, given that we have set our controller π and the input signal $\mathbf{x}$, the state $\mathbf{s}$ and the parameter $\boldsymbol{\theta}$ remain in their admitted sets. Supervisability is defined as:

$$S(P_i^j) := \Pr(\mathbf{q} \in Q | \mathbf{x} \in X_\mathrm{adm}, \mathbf{s} \in S_\mathrm{adm}, \boldsymbol{\theta} \in \Theta_\mathrm{adm}, T < T_\mathrm{max}). \quad (7)$$

where X_adm, S_adm, and Θ_adm are the sets of admitted values for $\mathbf{x}$, $\mathbf{s}$, and $\boldsymbol{\theta}$, respectively, and the throughput time T necessary for producing a product instance does not exceed T_max. S measures the maturity of a process only from the technical point of view. To determine whether the resulting optimized process P^* is economic, the cost of the process must also be considered. Applying the economic aspect is beyond the scope of this publication, but will be considered in future work.

4 Numerical example: electric arc furnace

We used an electric arc furnace (EAF) system modeled in [BLA03] as a numerical example to showcase our definitions. An EAF system is the steelmaking process that uses the heat of an electric arc to melt the iron. According to studies in the literature, in which three-phase EAF systems with three electrodes were modeled, the general model of the system is in this form $\mathbf{i}(\mathbf{s}_\mathrm{pos}, \mathbf{u}) : \mathbb{R}^3 \to \mathbb{R}^3$, where the inputs are the positions of each electrode in meters (m) and the outputs are the currents of each electrode in ampere (A). The relationship between current and position of each electrode is explained in more detail in the Appendix A.

4.1 Controller design

One of the quality features considered for the EAF system is the temperature. It is assumed that heat transfer to the liquid metal is fast enough so that the temperature of the liquid metal is uniform throughout [BCP99]. A similar situation applies to liquid slag.

There are differences in temperature between the melted part of the furnace, the scrap and the solid part of the slag. The melted part heats both the scrap and the solid part of the slag. The heating rate is proportional to the difference between the temperature of the solid part and the temperature of the liquid part.

We consider that the liquid part is heated by the heat it receives from the arc of the electrodes, so the formula for the rate of change of temperature is:

$$\frac{ds_{\text{temp}}}{dt} = \frac{Q_{arc}}{mc_p} \tag{8}$$

where s_{temp} is the temperature in Kelvin, and we want to achieve $1773\ \text{K}$ as the target temperature. Q_{arc} is the power received from the electrodes, m is the mass of the liquid steel, and c_p is the heat capacity of the liquid steel. According to [LDŠ12], $c_p = 0.84\ \text{KJ/KgK}$, and we assume $m = 80\ \text{t}$ and the temperature starts at $298\ \text{K}$ which is the temperature of the room at this temperature $c_p' = 0.45\ \text{KJ/KgK}$. We produce Q_{arc} by calculating the formulas below.

The formula for the power of all three electrodes is $Q_{arc} = \mathbf{u}^\top \mathbf{i}, \quad E_{arc} = \int_0^T Q_{arc} dt + E_{start}$, where $\mathbf{u}$ and $\mathbf{i}$ are vectors of RMS values of voltage and current of each electrode, and E_{start} is calculated with $E_{start} = mc_p' \Delta s_{\text{temp}} = 10.728\ \text{GJ}$, where $\Delta s_{\text{temp}} = 298\ \text{K}$. We know the eventual energy that we want to achieve from the eventual temperature that is our desired temperature. Considering the initial energy (E_{start}), which is the energy we have at room temperature ($298\ \text{K}$) we can calculate the difference energy ($\int_0^T Q_{arc} dt$) that we have to inject into the EAF system.

We consider that in the EAF model explained in the previous section, we have output noise. Therefore, we added sensor noise $\mathcal{N}(0\ \text{A}, 500\ \text{A}^2)$ to the measurement of the output current. In addition, we have initial condition noise $\mathcal{N}(0\ \text{m}, 0.1\ \text{m}^2)$ for the positions of the electrodes. Therefore, we solved the stochastic differential equation (SDE) for the relationship between the output current and the position of the electrodes:

$$d\mathbf{s}_{\text{pos}} \propto (\check{\mathbf{s}}_{\text{pos}} - \mathbf{s}_{\text{pos}}(t))dt + 0.05 dW_t \tag{9}$$

where W_t denotes a standard Brownian motion [GDHK11]. We designed a controller to achieve the desired product quality, i.e. a steel temperature of $1773\ \text{K}$. The input values that can be set are the positions of the electrodes. Furthermore, the electric current can be switched off once at an appropriate time (we do not allow bang-bang control – switching on/off repeatedly – during operations).

First, we compute the power required from the arc furnace to achieve the desired temperature after the duration T. This results directly from Equation (8). From this we compute $\mathbf{i}_{\text{des}}$, so that the target power is achieved and the maximum current that flows through any of the electrodes is minimal.

Second, recall that the mathematical model of the EAF predicts the current $\mathbf{i}$ resulting from the admissible positions of the electrodes $\mathbf{s}_{\text{pos}} \in S_{\text{adm}}$. The final position of the electrodes is selected as the solution of an optimization problem. This optimization problem is solved statically at the start of operations.

$$\min_{\mathbf{s}_{\text{pos}} \in S_{\text{adm}}} \|\mathbf{i}_{\text{des}} - \mathbf{i}(\mathbf{s}_{\text{pos}})\| \tag{10}$$

Here $\mathbf{i}_{\text{des}}$ is the vector of desired currents that we want to achieve for each electrode, and $\mathbf{i}(\mathbf{s}_{\text{pos}})$ is the vector of current of each electrode which is function of its position. The set of admissible values for the position of each electrode, S_{adm} is $[0\ \text{m}, 1.5\ \text{m}]^3$, as considered in [BLA03].

Third, a dynamic state-feedback controller at runtime tries to achieve the target position of the electrodes (accounting for any noise) and performs a controlled shut-off of the electric current when the desired temperature is achieved (under sensor noise for the temperature sensors).

4.2 Maturity quantification for the electric arc furnace system

Forcability plot

For calculating the Forcability considering the state of the system, which is position $\mathbf{s}_{\text{pos}}$ we have to calculate this probability:

$$F_\pi(T, \eta_{s^1_{\text{pos}}}) := \Pr(\|s^1_{\text{pos}}(T) - \breve{s}^1_{\text{pos}}\| < \eta_{s^1_{\text{pos}}} \,|\, s^1_{\text{pos}}(0), x(t) = \pi(Y_t), \breve{s}^1_{\text{pos}}) \tag{11}$$

It takes 10 s (ramp-up time) for the EAF to reach the desired current of 10 kA. Thus, the time duration is $T = 10$ s. As the calculation for position of all of the electrodes are the same we just illustrate the Forcability for the position of the first electrode s^1_{pos}, so $\eta_{s^1_{\text{pos}}} \in [0\text{ m}, 0.5\text{ m}]$ is the precision(error) that we want to achieve. $s^1_{\text{pos}}(0)$ has the form that we represent in Section 4.1 in the (SDE) so $s^1_{\text{pos}}(0) = 0\text{ m} + \mathcal{N}(0\text{ m}, 0.1\text{ m}^2)$, controller π is the controller for achieving the desired current which was explained in Section 4.1(9), and the equation (14) in the Appendix B, and considering the solution of the optimization problem (10), the desired position is $\breve{s}^1_{\text{pos}} = 0.79$ m.

To empirically calculate the probability, we run the simulation 1000 times. In each iteration, for each specific $\eta_{s^1_{\text{pos}}}$ we counted errors $\|s^1_{\text{pos}}(T) - \breve{s}^1_{\text{pos}}\|$ less than it at different durations $T \in \{0\text{ s}, 1\text{ s}, .., 10\text{ s}\}$. Finally, we divided the count value corresponding to each $\eta_{s^1_{\text{pos}}}$ for each specific duration T by 1000 to find the probability of having an error less than it. Figure 3 illustrates this Forcability plot.

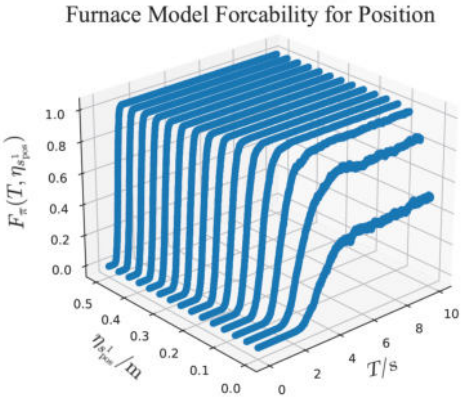


Figure 3: The Forcability plot for the position of the first electrode of the EAF model s^1_{pos} illustrates the probability of steering s^1_{pos} to its desired value with specific precision, as presented in (11).

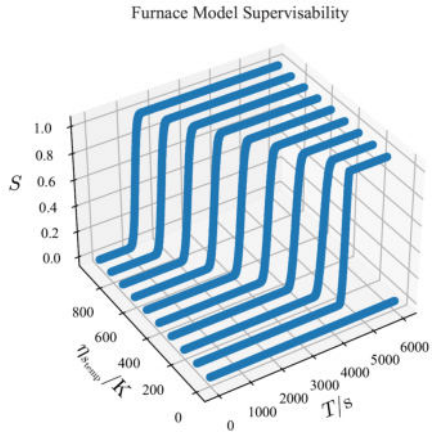


Figure 4: Supervisability plot for achieving the desired temperature as illustrated in equation (12).

As expected, essentially the probability increases with increasing error $\eta_{s^1_{\text{pos}}}$ and time T monotonically. The noises in the lines shows the uncertainty caused by sensor noise and the output noise of the system.

Supervisability plot

For calculating the Supervisability considering the temperature s_{temp} as the feature of quality we have to calculate the probability:

$$S := \Pr(\|s_{\text{temp}}(T) - \breve{s}_{\text{temp}}\| < \eta_{s_{\text{temp}}} \,|\, \mathbf{s}_{\text{pos}} \in [0\text{ m}, 1.5\text{ m}]^3, T < 10\text{ h}) \tag{12}$$

where the time duration $T = 6000$ s should be in the acceptable range which is $T < 10$ h, $\breve{s}_{\text{temp}} = 1773$ K is the desired temperature that we want to achieve at the end of time duration T, $\eta_{s_{\text{temp}}} \in [0\text{ K}, 1000\text{ K}]$ is the precision (error) that we want to achieve. $\mathbf{s}_{\text{pos}} \in [0\text{ m}, 1.5\text{ m}]^3$ represents the vector of electrode positions and should be in its admissible set $S_{\text{adm}} = [0\text{ m}, 1.5\text{ m}]^3$.

Estimating S is accomplished analogously to calculating the Forcability in the previous part, and we run the simulation 1000 times. In each iteration, for each specific $\eta_{s_{\mathrm{temp}}}$ we counted errors $\|s_{\mathrm{temp}}(T) - \breve{s}_{\mathrm{temp}}\|$ less than it at different durations $T \in \{0\ \mathrm{s}, 1\ \mathrm{s}, .., 6000\ \mathrm{s}\}$. Finally, we divided the count value corresponding to each $\eta_{s_{\mathrm{temp}}}$ for each specific duration T by 1000 to find the probability of having an error less than it. Figure 4 illustrates this Supervisability plot. As expected, the probability increases with increasing time and error $\eta_{s_{\mathrm{temp}}}$ monotonically.

5 Summary and outlook

In this paper, we introduced new definitions to quantify the maturity of a production process. We introduced Elucidability and Forcability which are generalization of the classical definitions in control theory: observability and controllability. E quantifies the ability to observe the states and parameters of the system, while F quantifies the ability to steer the outputs and states of the system, which are related to the internal dynamics of the system. S represents the probability of achieving the desired quality for the final product. Using the example of an EAF system, we illustrate E, F, and S in a practical example. The presented probability plots can be compared to assess how changes in the process can increase its maturity. In the future, we plan to investigate the impact of hardware and software changes on E, F, and S and to explore whether there is a monotonic relationship between these quantities.

Additionally, Figure 2 illustrates an idealized version of the closed-loop automated process instance, excluding networks of processes or manual processes with human intervention. In the future, we can explore distributed production processes, where each part may be located in different geographic locations, as well as processes that involve human intervention, decision-making, and other human factors. Moreover, taking into account the increased complexity of the system, especially when attempting to calculate S in a system where quality metrics $\mathbf{q}(p)$ are subjective or context-dependent.

Acknowledgments

This project is funded by DFG FOR 5339.

References

[BCP99] Johannes Gerhardt Bekker, Ian Keith Craig, and Petrus Christiaan Pistorius. Modeling and simulation of an electric arc furnace process. *ISIJ international*, 39(1):23–32, 1999.

[Bey99] Jürgen Beyerer. The value of additional knowledge in measurement–a bayesian approach. *Measurement*, 25(1):1–7, 1999.

[BG16] Jürgen Beyerer and Jürgen Geisler. A framework for a uniform quantitative description of risk with respect to safety and security. *European Journal for Security Research*, 1:135–150, 2016.

[BLA03] Benoit Boulet, Gino Lalli, and Mark Ajersch. Modeling and control of an electric arc furnace. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 4, pages 3060–3064. IEEE, 2003.

[FSL03] J Fleischer, D Spath, and G Lanza. Quality simulation for fast ramp up. In *Proceedings of the 36th CIRP International Seminar on Manufacturing Systems*, 2003.

[GDHK11] HP Geering, G Dondi, F Herzog, and S Keel. Stochastic systems. *Course script*, 2011.

[HN89] AMA Hamdan and AH Nayfeh. Measures of modal controllability and observability for first-and second-order linear systems. *Journal of guidance, control, and dynamics*, 12(3):421–428, 1989.

[HS80] PC Hughes and RE Skelton. Controllability and observability of linear matrix-second-order systems. *Journal of applied mechanics*, 47(2):21–39, 1980.

[Kal60] Rudolf E Kalman. On the general theory of control systems. In *Proceedings first international conference on automatic control, Moscow, USSR*, pages 481–492, 1960.

[Kal63] Rudolf Kalman. Controllability of linear dynamical systems. *Contributions to differential equations*, pages 189–213, 1963.

[KPPS09] Okhyun Kang, Youngjin Park, YS Park, and Moonsuk Suh. New measure representing degree of controllability for disturbance rejection. *Journal of guidance, control, and dynamics*, 32(5):1658–1661, 2009.

[LDŠ12] Vito Logar, Dejan Dovžan, and Igor Škrjanc. Modeling and validation of an electric arc furnace: Part 1, heat and mass transfer. *ISIJ international*, 52(3):402–412, 2012.

[LP16] Haemin Lee and Youngjin Park. Degree of controllability for linear unstable systems. *Journal of Vibration and Control*, 22(7):1928–1934, 2016.

[Met11] Tobias Mettler. Maturity assessment models: a design science research approach. *International Journal of Society Systems Science*, 3(1-2):81–98, 2011.

[MKG02] Benoît Marx, Damien Koenig, and Didier Georges. Optimal sensor/actuator location for descriptor systems using lyapunov-like equations. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 4, pages 4541–4542. IEEE, 2002.

[MKG04] Benoît Marx, Damien Koenig, and Didier Georges. Optimal sensor and actuator location for descriptor systems using generalized gramians and balanced realizations. In *Proceedings of the 2004 American Control Conference*, volume 3, pages 2729–2734. IEEE, 2004.

[MMSM13] Eileen McConkie, Thomas A Mazzuchi, Shahram Sarkani, and David Marchette. Mathematical properties of system readiness levels. *Systems Engineering*, 16(4):391–400, 2013.

[MW72] PC Müller and HI Weber. Analysis and optimization of certain qualities of controllability and observability for linear dynamical systems. *Automatica*, 8(3):237–246, 1972.

[PCCW93] Mark C Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V Weber. Capability maturity model, version 1.1. *IEEE software*, 10(4):18–27, 1993.

[Pfl95] Shari Lawrence Pfleeger. Maturity, models, and goals: How to build a metrics plan. *Journal of Systems and Software*, 31(2):143–155, 1995.

[SH05] Abhay K Singh and Juergen Hahn. Determining optimal sensor locations for state and parameter estimation for stable nonlinear systems. *Industrial & engineering chemistry research*, 44(15):5645–5659, 2005.

[SH06] Abhay K Singh and Juergen Hahn. Sensor location for stable nonlinear dynamic systems: Multiple sensor case. *Industrial & engineering chemistry research*, 45(10):3615–3623, 2006.

[SLM17] Hamid Reza Shaker and Sanja Lazarova-Molnar. A new data-driven controllability measure with application in intelligent buildings. *Energy and Buildings*, 138:526–529, 2017.

[SM96] Hossein Saiedian and Laura M McClanahan. Frameworks for quality software process: Sei capability maturity model versus iso 9000. *Software Quality Journal*, 5:1–23, 1996.

[SS+04] Catherine Soanes, Angus Stevenson, et al. *Concise oxford English dictionary*, volume 11. Oxford university press Oxford, 2004.

[ST24] Kazuhiro Sato and Shun Terasaki. Controllability scores for selecting control nodes of large-scale network systems. *IEEE Transactions on Automatic Control*, 2024.

[Tar92] M Tarokh. Measures for controllability, observability and fixed modes. *IEEE Transactions on Automatic Control*, 37(8):1268–1273, 1992.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[VLL84] CN Viswanathan, RW Longman, and PW Likins. A degree of controllability definition-fundamental concepts and application to modal systems. *Journal of Guidance, Control, and Dynamics*, 7(2):222–230, 1984.

[WK02] Thomas Wettstein and Peter Kueng. A maturity model for performance measurement systems. *WIT Transactions on Information and Communication Technologies*, 26, 2002.

[XYC+18] Yaping Xia, Minghui Yin, Chenxiao Cai, Baoyong Zhang, and Yun Zou. A new measure of the degree of controllability for linear system with external disturbance and its application to wind turbines. *Journal of Vibration and Control*, 24(4):739–759, 2018.

A EAF detail

Figure 5 is the physical model of the EAF system [BLA03]. The relation between the position

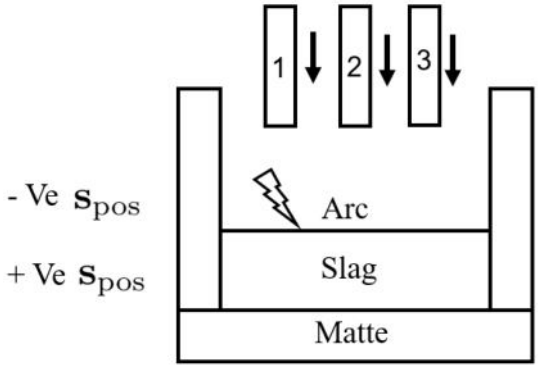


Figure 5: The illustration of three phase EAF system [BLA03]

and the current of each electrode is:

$$\mathbf{i} = \mathbf{G}(\mathbf{s}_{\mathrm{pos}}, \mathbf{u})\mathbf{s}_{\mathrm{pos}} + \mathbf{b}(\mathbf{s}_{\mathrm{pos}}, \mathbf{u})$$

$\mathbf{i}$ is a 3×1 vector shows the current of each electrode, $\mathbf{s}_{\mathrm{pos}}$ is a 3×1 vector shows the position of each electrode, $\mathbf{G}$ is the 3×3 conductance matrix, and $\mathbf{b}$ is the 3×1 constant vector and both are function of voltage and position. g_k denotes the slag to matte conductance and its relation with the position of each electrode is:

$$g_k = c_k s^k_{\mathrm{pos}} + g_s$$

where $k = 1, 2, 3$ denotes the three different electrodes, and c_k is the conductance coefficient (in siemens (S)/m)), s^k_{pos} is the position of each electrode (in m) and g_s is the conductance of the slag when the electrodes are positioned at the surface of the slag (in S). In modeling the system from the physical model to the electric circuit, the matte is considered the virtual ground, so the voltage of this node in the electric circuit $u_m = 0$ V. In addition, the inter-electrode conductance is considered the same for all three electrodes and is denoted by g. The dynamic equation of the system is:

Table 1: Table of variables in the EAF system, we reproduce the values from the original paper

Variables	Acceptable Range	Chosen Value
u_1, u_2, u_3	$100 - 1000$ V	500 V
c_1, c_2, c_3	$1 - 100$ S/m	20 S/m
g_s	$5 - 25$ S	10 S
g	$[0, 0.15]$ S	0.1 S

$$\mathbf{G} = \frac{1}{g_{tot}} \begin{bmatrix} 2u_1c_1(g_s+g) - u_2c_1(g_s+g) & u_1c_2(g_s+2g) - u_2c_2(g_s+g) & u_1c_3(g_s+2g) - u_2c_3g \\ -u_3c_1(g_s+g) + c_1c_2s^2_{\text{pos}}(u_1-u_2) & -u_3c_2g & -u_3c_3(g_s+g) \\ +c_1c_3s^3_{\text{pos}}(u_1-u_3) & & \\ -u_1c_1(g_s+g) + u_2c_1(g_s+2g) & -u_1c_2(g_s+g) + 2u_2c_2(g_s+g) & -u_1c_3g - u_2c_3(g_s+2g) \\ -u_3c_1g & -u_3c_3(g_s+g) + c_2c_3s^3_{\text{pos}}(u_2-u_3) & -u_3c_3(g_s+g) \\ & +c_1c_2s^1_{\text{pos}}(u_2-u_1) & \\ -u_1c_1(g_s+g) - u_2c_1g & -u_1c_2g - u_2c_2(g_s+g) & -u_1c_3(g_s+g) - u_2c_3(g_s+g) \\ -u_3c_1(g_s+2g) & +u_3c_2(g_s+2g) & 2u_3c_3(g_s+g) + c_1c_3s^1_{\text{pos}}(u_3-u_1) \\ & & +c_2c_3s^2_{\text{pos}}(u_3-u_2) \end{bmatrix}$$

$$\begin{bmatrix} i_1 \\ i_2 \\ i_3 \end{bmatrix} = \mathbf{G} \begin{bmatrix} s^1_{\text{pos}} \\ s^2_{\text{pos}} \\ s^3_{\text{pos}} \end{bmatrix} + \frac{g_s^2 + 3gg_s}{g_{tot}} \begin{bmatrix} 2u_1 - u_2 - u_3 \\ -u_1 + 2u_2 - u_3 \\ -u_1 - u_2 + 2u_3 \end{bmatrix} \tag{13}$$

where the u_1, u_2, and u_3 are the voltage of each electrode and $g_{tot} = c_1 s^1_{\text{pos}} + c_2 s^2_{\text{pos}} + c_3 s^3_{\text{pos}} + 3g_s$. Table 1 shows the typical values for this EAF system as a three-phase electric circuit. The voltages are phasors with 500 V amplitude and 120° phase difference, so for each electrode we have $u_1 = u_2 = u_3 = 500$ V, $c_1 = c_2 = c_3 = 20$ S/m, $g_s = 10$ S and $g = 0.1$ S.

B Dynamic equation of the system

The dynamic equation for the system is related to the relation of the position and velocity of the electrodes:

$$\frac{d\mathbf{s}_{\text{pos}}}{dt} \propto \mathbf{s}_{\text{pos}}(t)$$

where $\mathbf{s}_{\text{pos}}$ (vector of the positions of the electrodes in meter) is the state of the system.

For controlling the system we consider that we have a state-feedback controller, so the dynamic equation of the closed loop system is in the form (ODE):

$$d\mathbf{s}_{\text{pos}} \propto (\breve{\mathbf{s}}_{\text{pos}} - \mathbf{s}_{\text{pos}}(t))dt$$

where $\breve{\mathbf{s}}_{\text{pos}}$ is the desired set point. Considering the noise that accrues in the system instead of (ODE) we will have a (SDE) of the form:

$$d\mathbf{s}_{\text{pos}} \propto (\breve{\mathbf{s}}_{\text{pos}} - \mathbf{s}_{\text{pos}}(t))dt + 0.05dW_t \tag{14}$$

where W_t denotes a standard Brownian motion [GDHK11].

WORKSHOP REPORT: LEARNING APPROACHES FOR HYBRID DYNAMICAL SYSTEMS

Swantje Plambeck[1], **Nemanja Hranisavljevic**[2], **Maximilian Schmidt**[1], **Kaja Balzereit**[3], **Aaron Bracht**[1], **Magnus Redeker**[4], **Negar Arabizadeh**[5], **Alexander Diedrich**[2], **Jens Eickmeier**[6], **Oliver Niggemann**[2], **and Goerschwin Fey**[1]
[1]Institute of Embedded Systems, Hamburg University of Technology, Hamburg, Germany
[2]Institute of Automation Technology, Helmut Schmidt University, Hamburg, Germany
[3]Bielefeld University of Applied Sciences and Arts, Bielefeld, Germany
[4]Fraunhofer IOSB-INA, Lemgo, Germany
[5]Karlsruhe Institute of Technology, Vision and Fusion Laboratory (IES), Karlsruhe, Germany
[6]SOLLICH KG, Bad Salzuflen, Germany

ABSTRACT

This report summarizes the workshop on "Learning Approaches for Hybrid Dynamical Systems", held at the 2025 Conference on Machine Learning for Cyber-Physical Systems (ML4CPS). The workshop aimed to strengthen collaboration and foster exchange between institutions engaged in research on model learning methods for hybrid CPSs.

The participating research groups approach the topic from diverse perspectives, for example, from an application perspective, from a tool perspective, or from a fundamental and formal perspective. Accordingly, this paper synthesizes the discussions from the workshop and presents an overview of key perspectives on several central topics, including the taxonomy of hybrid systems, current learning paradigms and techniques, and particularly representative use cases.

Keywords Cyber-Physical Systems, Hybrid Dynamical Systems, Model Learning, Model Inference, Hybrid Automata, Model Interoperability

1 Introduction

By integrating discrete and continuous dynamics, hybrid systems can effectively represent real-world systems in various domains, including production engineering [20], biological applications [4], or automotive technology [3]. As a result, they have emerged as a widely recognized and important modeling formalism.

The wide range of applications has led to the development of numerous modeling approaches and formalisms for describing hybrid systems. This diversity fosters open discussion and exchange within the community regarding differing perspectives on hybrid systems and their modeling. The workshop "Learning Approaches for Hybrid Dynamical Systems", held at the 2025 Conference on Machine Learning for Cyber-Physical Systems (ML4CPS), leveraged the expertise of individual research groups to bring together these diverse perspectives and approaches. This paper summarizes the key discussions from the workshop and offers an overview of current viewpoints as well as initial ideas for addressing open challenges in the field.

This article is organized around several key topics discussed during the workshop:

1. **Taxonomy of hybrid systems**: As a first step, a common taxonomy is developed to establish a shared understanding across different perspectives on hybrid systems. The taxonomy is structured around key aspects, including the representation of continuous dynamics, the representation of discrete dynamics, and the modeling of transitions between discrete modes.
2. **Learning of hybrid dynamical systems**: The second topic addresses the learning of hybrid dynamical systems, distinguishing different learning sub-steps and highlighting the substantial challenges posed by inherent complexity and the mixed nature of discrete and continuous dynamics.
3. **Real-world use cases**: An additional contribution is the presentation of two real-world use cases. For one, the observed data are publicly available, while the second – although the data are confidential – represents a complex, real-world CPS with several unresolved challenges.
4. **Integration into industry**: Finally, the discussion turns to asset administration shells, which serve as model characterizations that enable the integration of machine learning models into industrial practice.

2 Topics Covered

2.1 Taxonomy of Hybrid Dynamical Systems

Initial Framing and Understanding

This section explores modeling and learning approaches for hybrid dynamical systems, determining a fundamental taxonomy that represents existing methodologies and applications and offers insights into open topics and new directions in the research field. For this, the taxonomy covers different aspects of the learning process individually, which are the system, the learner, and the model. The system is the real-world system that is to be modeled, the learner is the algorithm that is used to learn the model, and the model is the abstract representation of the system.

System to Model: Abstraction and Preprocessing

Often, a system under learning is considered a black box, and the learning process aims to derive a detailed representation and understanding of the inner workings of the system. We might assume that the system is a hybrid automaton and search for the corresponding hybrid automaton model. This representation is usually an abstraction of the real-world system. Such an abstraction involves, for example, the following aspects:

- Discretization: While the system is continuous, the model or part of the model might be discrete [21], [22]. For hybrid automata, these discrete parts are the modes and transitions. Techniques like Variational Autoencoders (VAE) [18, 21], Boltzmann machines, Principal Component Analysis (PCA), Clustering [26], and Self-Organizing Maps (SOMs) [21], are possible options for performing a discretization.
- Unobservable Variables: The system might have unobservable variables that influence the output of the system. These variables should be considered in the model.
- Inputs & Outputs: The signals of the system need to be mapped to the inputs and outputs of the model or defined as such. In many cases, this mapping is straightforward, but in some cases, the searched model might require a different mapping.
- Modeling of Subsystems: The model might only consider parts of the system. This also includes leaving out real-world effects such as noise or other disturbances.
- Maturity of the Model: The model might be more or less mature or efficient. Maturity quantifies, for example, the ability of the system to estimate unobservable states or to

steer the output signal (through the input signal) to reach and stabilize at its desired value, even in the presence of noise and disturbances [2].

- Further Preprocessing: Any kind of preprocessing, such as de-noising, filtering, and feature extraction, contributes to an abstraction between the system and the model [16].

While there exist various techniques to perform these abstractions and examples of their applications, there are still several new methods as well as combinations of existing methods to be explored. A relevant aspect discussed in this regard is the re-usability of models across different applications. From a domain-overlapping perspective, this can be the unification of models that serve specific purposes, e.g., all predictive models, or all models that are used for testing should be based on similar modeling paradigms.

Considering a fixed domain and application, model re-usability can be achieved by altering the inputs and outputs of the model. Thus, instead of considering a model as a fixed direction for the transfer of signals, the model is a flexible tool where all external signals can serve both for input and output. This is a challenging approach, especially since the behavioral representation of the model is invertible.

A further challenge are models for explanation. These models might need to be altered on the basis of the explanations sought. This goes beyond the mere alteration of inputs and outputs, as the representation of inputs and outputs of the model might need to be adapted as well.

System to Learner: Complexity Dimensions

The relation between system and model is further influenced by the learner, which usually defines which type of model is learned, e.g., a hybrid automaton, a neural network, or a decision tree. The learner is the interface between the system and the model. Thus, the learner on the input side must match the complexity of the system and the provided learning data, and on the output side must provide a model that is able to represent the system. We first discuss the complexity dimensions of the system and then continue with the type of model that is learned.

The complexity of a system is hard to quantify as it is influenced by various factors [6]. The following list gives an overview of the most important dimensions that influence the complexity of a system, while we consider a hybrid automaton representation of the system:

- Number of Variables: The number of variables that are considered in the system, including inputs, outputs, and states.
- Set of Possible Flow Functions: The set of possible flow functions that can be used in the system. This set can be restricted, e.g., to linear differential equations or polynomial functions. A further subdimension is, for example, the order of the differential equation or polynomial function.
- Number of Modes: The number of modes that are considered in the system. This can be a single model or a multi-model structure. Often, there is a trade-off between the number of modes and the complexity of the flow functions. If the flow functions are simple, more modes are needed to represent the system. In case of complex flow functions that might represent jump conditions, this could aggregate multiple modes in a single one [23].
- Conditions and Jumps: The conditions and jumps in the transition events. Transition events can be defined in various ways, e.g., as a condition on the state variables or as a stochastic event. The complexity of the system is influenced by the number of conditions and jumps that are considered, but also by the complexity of these conditions and jumps.
- Stochasticity: As mentioned with the transition events, these might be stochastic. In practice, such determinism might be caused by noise, uncertainty, or unobservable variables. Often, a stochastic system is considered more complex because the behavior is harder to predict. However, introducing stochasticity can also simplify the model, as it might be easier to represent the system with fewer modes, simpler flow functions,

or fewer conditions. A maturity quantification of the model could give insights in the ability to achieve modeling goals under stochastic conditions [2].

The final dimensions of this list of complexity dimensions suggest that the handling of noise and uncertainty is a particular challenge in the learning process. Especially for continuous dynamic, a complete coverage of the state space on final learning data sets is impossible. Thus, the model must be able to generalize from the data that is available, but at the same time, it must be able to handle the noise or ambiguity that is present in the system. In the optimal case, the model is able to distinguish between noise and real system behavior and to provide uncertainty estimates for the model itself.

A complexity dimension that goes beyond single-system models is multi-model structures. Again, a multi-model structure is often considered more complex, as it requires the coordination of multiple models. Nevertheless, a multi-model structure can also simplify the representation of the system, as it allows for a more modular representation of the system. This can be especially useful if the system is composed of multiple subsystems that can be represented by different models. These models might even be of different types to specifically adapt to the subsystem that is modeled.

Learner to Model: Expressiveness of the Result

In model learning for hybrid dynamical systems, the learner produces a model that captures the behavior of the underlying system. Typically, the type of model is predetermined by the learning method. Depending on the learning strategy, the model can be learned directly from the data or inferred from the observed behavior of the system [23].

Learning directly from data constructs model components such as system modes, continuous dynamics, and transitions based on input-output traces or time-series data. In contrast, inferring a model from behavior involves reasoning about system structure and dynamics based on observed properties or execution traces, often guided by prior knowledge or structural constraints [23].

Suppose that the system is represented by a hybrid automaton, the learning process can take several forms. The hybrid automaton may be learned explicitly: its discrete modes, flow functions, and transition logic are directly extracted from the data. Alternatively, the model may be only partially explicit. For example, the mode structure might be identified from the data, while the continuous dynamics (i.e., flow functions) are approximated using regression-based methods such as neural networks or symbolic regression [19]. This introduces a dependency on the abstraction between the physical system and its model representation. When the predefined model class restricts the allowable flow functions, it may necessitate approximating or abstracting the original dynamics to fit the target model.

A further approach involves using purely data-driven regression models to capture system behavior without constructing an explicit hybrid automaton. In such cases, methods with high representational capacity—such as neural networks, kernel methods, or support vector machines—can effectively approximate system trajectories, even if the resulting model lacks an interpretable hybrid structure.

The degree to which a model is explicit, partially explicit, or inferred has significant implications for its expressiveness, interpretability, and explainability. When the learned model mirrors the hybrid automaton structure of the true system, it can offer insights into system dynamics, such as physical laws and variable dependencies—features that are valuable for applications like system diagnosis. In contrast, non-explicit models are often simpler to train and may offer advantages in terms of computational efficiency.

Explicit learning of hybrid automata also offers structural modularity: the model consists of three interconnected elements: (1) discrete modes, (2) continuous flow functions, and (3) transition relations. This separation allows different learning techniques to be applied to each component, offering high flexibility. The benefits of this modular pipeline approach are discussed further in Section 2.2.

Finally, the learner and the selected model class also define the model complexity dimensions. Some of these dimensions are externally defined (e.g., the class of admissible flow functions),

while others must be inferred during learning (e.g., the number of modes). The learner must be capable of adapting to the true complexity of the system, which can be particularly challenging if the complexity is unknown a priori or lies outside the expressiveness of the chosen model class.

2.2 Learning Process for Hybrid Systems

Initial Framing and Understanding

Learning hybrid automata is a challenging task due to the interplay between discrete and continuous dynamics. In the literature, the learning process is often structured into modular steps to incrementally build a comprehensive model of the system. Figure 1 illustrates a commonly adopted learning pipeline, which consists of the following steps:

1. **Trace Segmentation**
   Identification of mode switches in observed system trajectories.
2. **Segment Clustering**
   Grouping of segments that exhibit similar dynamic behavior, assumed to correspond to identical dynamical modes.
3. **Mode Characterization**
   Derivation of continuous flow functions that describe the behavior within each identified mode.
4. **Transition Identification**
   Determination of transition conditions or guard predicates that govern the switching behavior between modes.
5. **Model Inference**
   Integration of the identified components—modes, transitions, and flow functions—into a unified hybrid automaton model.

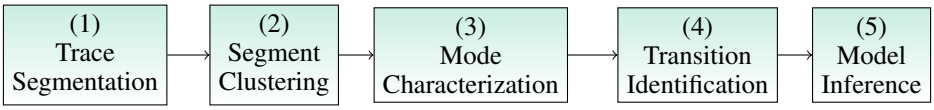


Figure 1: Standard learning process for hybrid automata

This modular formulation allows flexibility in selecting different techniques at each stage, as summarized in Table 1. Some methods—such as Self-Organizing Maps or Decision Trees—can be employed at multiple steps, promoting tighter integration and efficient information flow across the learning pipeline. To select a suitable technique, maturity quantification is a possible base to compare the efficiency of different techniques used in the learning process (such as their ability to steer the output signal within a specific error margin, estimate unobservable variables with a certain precision, or ultimately achieve the desired product quality after implementing the learned model) [2].

Although this structure is widely adopted, it implicitly assumes a linear and decoupled progression through the steps. However, this assumption is not without limitations. In particular, the segmentation of traces is often performed without knowledge of the actual system dynamics, which contradicts the very rationale for identifying distinct modes—namely, to represent distinct types of dynamical behavior that cannot be captured by a single flow function.

Current approaches frequently use purely statistical or geometric criteria for segmentation, such as Euclidean distance or changepoint detection, which may not align with the true dynamical boundaries of the system. This can lead to an unnecessary large number of segments and modes, resulting in models that are more complex than required and potentially harder to interpret, generalize, or verify.

Trace Segmentation	Euclidean Distance, Self-Organizing Maps, Changepoint Detection, . . .
Segment Clustering	Dynamic Time Warping, Self-Organizing Maps, Clustering, . . .
Mode Characterization	Linear Regression, Self-Organizing Maps, Linear Matrix Inequalities, . . .
Transition Identification	Condition Mining, Rule-Based Systems, Decision Trees, . . .
Model Inference	Automata Learning, Decision Trees, Petri Net Construction, . . .

Table 1: Techniques Applied in the Learning Process

Revisiting the Learning Pipeline: Toward Model-Guided Segmentation

A key insight is that multiple modes are only necessary when a single flow function cannot sufficiently capture the observed behavior. Therefore, we argue that trace segmentation should not be an independent, upstream step but rather be informed by the process of mode characterization. That is, segmentation decisions should be governed by the expressiveness of the modeling technique used to characterize each mode.

Concretely, instead of predefining mode boundaries, one could attempt to model the entire trace using a candidate flow function (e.g., linear regression, neural networks, etc.). Where the modeling error becomes unacceptably high—indicating that the current function cannot represent the underlying behavior—a segmentation point is introduced. This approach couples segmentation with model expressiveness, leading to models that introduce new modes only when necessary.

This shift from purely data-driven segmentation to model-informed segmentation would reduce unnecessary model complexity. Moreover, it opens the possibility for iterative refinement, where segmentation and mode characterization co-evolve through successive approximations, improving both segmentation accuracy and overall model quality.

Implications and Future Directions

This perspective challenges the rigid step-by-step pipeline by advocating for integrated and feedback-driven learning processes. For example, segment clustering can be refined based on insights from mode characterization; similarly, transition conditions might inform whether two clusters should be merged or split. Such iterative and adaptive processes stand in contrast to the traditional linear approach and better reflect the intertwined nature of discrete and continuous system behavior.

Future work should explore algorithmic frameworks that implement this vision—integrating model expressiveness into early-stage segmentation, and promoting end-to-end learning pipelines with dynamic feedback loops between stages. By making segmentation contingent on representational limits, we expect to achieve models that are both simpler and more faithful to the underlying system dynamics.

2.3 Use cases and datasets

Initial Framing and Understanding

Two specific use cases were examined to stimulate discussion, highlight key challenges, and explore preliminary solution approaches.

Conveyor System from SF-OWL

The conveyor system is part of the high-rack storage system from SmartFactoryOWL, Lemgo, Germany. It consists of six belt conveyors (the state as in year 2016 when the data were recorded) [5]. The dataset is available online at `https://www.kaggle.com/datasets/inIT-OWL/high-storage-system-data-for-energy-optimization`.

The system consists of four horizontal conveyors: LH1, LH2, RH1 and RH2 and two vertical conveyors: LV and RV, which move LH2 and RH2, respectively. While the motion of vertical conveyors is controlled to achieve the set-point vertical position, horizontal conveyors stop (and run) based on the binary presence sensors detecting the horizontal position of an item. There are four operations (movements) that the conveyor system can perform (see Figure 2).

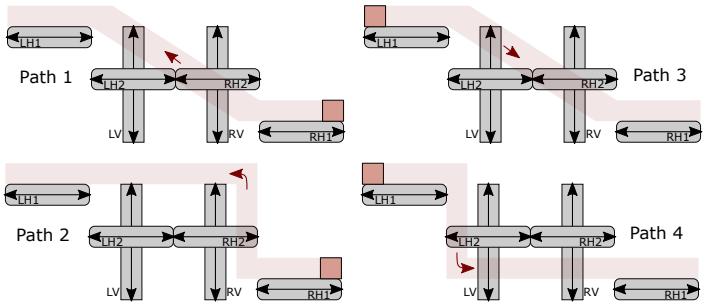


Figure 2: The conveyor system from SmartFactoryOWL.

This is a hybrid system consisting of 6 discrete events and 24 continuous variables. Event variables: the electrical drives controlling the conveyors receive from the PLC different commands which can take one of the two (in case of vertical conveyors) or three values (in case of the horizontal conveyors). The 24 continuous variables represent position, power, voltage, and current of each of the six conveyors. In total, 49830 observations in two files (sequences) are recorded during periods of 16.3 and 35.5 minutes, respectively. In both files, the system was alternating the following sequence of operations: Path1, Path3, Path2, Path4, Path1, Path3,...

Columbus ECLSS dataset – project KISS ([1])

The Environmental Control and Life Support System (ECLSS) is a CPS on board the COLUMBUS module of the International Space Station (ISS). ECLSS is tasked with maintaining habitable conditions for astronauts through a number of valves, heat exchangers, pipes, etc. and multiple control loops. At the same time, it transmits telemetry data to an Earth-based location at one second intervals to facilitate operational support. The support provided by a human operator can react to different situations and, if necessary, can reconfigure the system in order to continue the system's functionality. As this process is currently based on the manually defined rules and the operator's understanding of the system state based on the telemetry data, an AI-based assistance could be highly beneficial.

The reconfiguration problem requires an algorithmic solution that can be executed at some point in time and return a set of recommended reconfiguration possibilities. The input to the reconfiguration algorithm is:

- a set of configuration constraints that constrain a subset of variables to have specific values (for example, a variable must be 0); and,
- observed data before the reconfiguration is triggered (e.g., during last 15 minutes).

The historical data to train the system model (available in the *KISS* project) is a substantial telemetry data set that covers observations of several hundreds of variables over 6 years, from the beginning of 2018 to the end of 2023.

The identified challenges are the following.

- Reconfiguration: The workshop explored the unique needs associated with the application of the hybrid system model in reconfiguration scenarios.
- Within the dataset, configurations are not uniformly represented, with the system predominantly functioning in a limited set of common configurations.
- Many valid configurations are absent from the dataset, for example, redundancy scenarios that are never used. Key question: Can we infer the behavior of the never-occurring modes?
- The system appears to be considerably complex due to numerous variables, hidden variables, and the extensive interaction between its discrete and continuous components [6].
- A very interesting challenge would be the creation of a common model that can be used for anomaly detection, diagnosis, and system reconfiguration. Currently, the KISS project proposes three distinct models for these tasks [14, 15, 17, 25].

2.4 Interoperable ML datasets and applications

The concept of the Asset Administration Shell (AAS) plays a pivotal role in the realm of Industrie 4.0 (I4.0), which is characterized by the intelligent interconnection of machines and processes through information and communication technology, [10, 12, 13]. As traditional value chains evolve into flexible, dynamic, and globally connected value networks, AAS facilitates the implementation of digital twins and ensures interoperability among the solutions of various suppliers.

A specific submodel within the AAS framework is the Artificial Intelligence (AI) Dataset submodel, [7], which is designed to manage and standardize information pertaining to datasets used in AI applications. The AI Dataset submodel finds its utility in various use cases, particularly in documenting datasets to simplify their reuse by providing standardized access to crucial meta-information, such as the conditions under which the data was generated. Moreover, the submodel enriches the management of the dataset by documenting static parameters such as mean values and data formats, as well as dynamic parameters such as environmental conditions affecting data collection, like humidity and temperature. In terms of functionality, the AI Dataset submodel allows data scientists and AI experts to access comprehensive dataset information and select suitable data for AI training based on provided metrics. It also supports the investigation of data-related issues and facilitates easy communication between users and creators through detailed contact information.

Beyond, the Artificial Intelligence Model Nameplate submodel provides a standardized structure for documenting AI models, facilitating efficient management and reuse of trained models while ensuring clear communication of model properties among stakeholders, [9], the Artificial Intelligence Deployment submodel focuses on the standardized deployment of AI models, particularly in industrial settings, enabling easier usage and management of the AI lifecycle, [8], and within the Capability submodel the capabilities and requirements of AI services can be described, [11].

For example, the integration of data-driven services in manufacturing environments is increasingly essential to improve efficiency and optimize operations. As shop floors become populated with several thousand machines, managing these machines and their corresponding value creation processes becomes overwhelmingly complex for humans. This complexity makes it challenging to track which machines are benefiting from existing data-driven services, how these services can be extended to additional machines, and whether different data-driven services can be combined to create greater value for specific machines.

To address these challenges effectively, a data-driven service can be equipped with an AAS that specifies its capabilities, the data required to utilize those capabilities, and the data or value that the service generates for machines, processes, or products. As depicted in Figure 3, an automatization of the matching of machines and data-driven services enables manufacturers to make informed decisions without being overwhelmed by the sheer volume of machines and services [24]. Ultimately, machines and data-driven services equipped with interoperable AASs

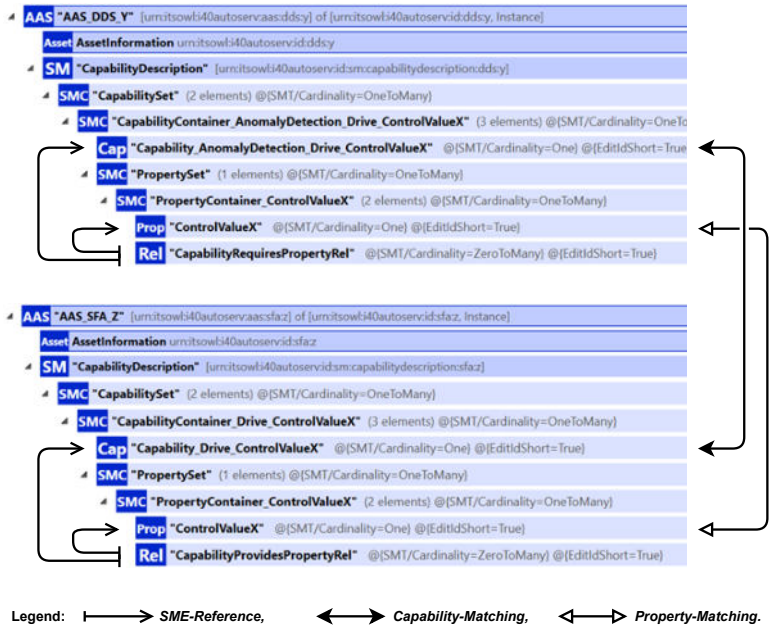


Figure 3: The matching of data-driven services (DDSs) and shop-floor assets (SFAs) like machines: Matching of their capabilities as well as required and provided properties based on AAS-specified capability descriptions. © Fraunhofer IOSB-INA.

not only enhance operational efficiency but also drive greater value creation on the shop-floor in a future-proof manner.

3 Conclusions and Outlook

In this paper, we dicuss the challenges and opportunities in learning hybrid dynamical systems. The development of learning approaches for hybrid dynamical systems is a crucial area of research. By properly addressing abstraction techniques, model complexities, and learning processes, we significantly enhance the understanding and application of these systems. We present a taxonomy that categorizes different modeling approaches and highlights the importance of understanding the underlying characteristics of the system, the learner, and the model. We also propose a comprehensive learning procedure for hybrid systems, which consists of five steps: trace segmentation, segment clustering, mode characterization, transition identification, and model inference. This modular approach allows for the application of various techniques at each step, enabling efficient information transfer and iterative refinement of the model. Further, we introduce a real-world data set from a complex Cyber-Physical System (CPS) and outline the learning process, emphasizing the need for flexibility and adaptability in extracting information from data. We discuss asset administration shells as a tool to put machine learning models into practice. With this report of the workshop on ”Learning Approaches for Hybrid Dynamical Systems” we, thus, present a diverse view on the current state of the art as well as open challenges and opportunities.

Acknowledgments

The authors thank the participants of the workshop for their valuable contributions and discussions. In particular, we would like to thank the organizers of the ML4CPS conference for providing a platform for this workshop.

References

[1] DTEC - (K)ISS – Künstliche Intelligenz für die Diagnose der ISS, 2022.

[2] N. Arabizadeh, J. Pfrommer, and J. Beyerer. How to quantify the maturity of production processes. In *ML4CPS–Machine Learning for Cyber-Physical Systems*. UB HSU, 2025.

[3] M. S. Branicky. Introduction to hybrid systems. In *Handbook of networked and embedded control systems*, pages 91–116. Springer, 2005.

[4] R. Ghosh and C. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *IEE Proceedings-Systems Biology*, 1(1):170–183, 2004.

[5] N. Hranisavljevic, A. Maier, and O. Niggemann. Discretization of hybrid CPPS data into timed automaton using restricted Boltzmann machines. *Engineering Applications of Artificial Intelligence*, 95:103826, 2020.

[6] N. Hranisavljevic, T. Westermann, S. Plambeck, H. S. Steude, G. Benndorf, and O. Niggemann. A model learning perspective on the complexity of cyber-physical systems. In *Conference on Machine Learning for Engineering (ML4CPS)*, 2025.

[7] Industrial Digital Twin Association e. V. Artificial Intelligence Dataset Submodel - IDTA Number: 02058.

[8] Industrial Digital Twin Association e. V. Artificial Intelligence Deployment Submodel - IDTA Number: 02059.

[9] Industrial Digital Twin Association e. V. Artificial Intelligence Model Nameplate Submodel - IDTA Number: 02060.

[10] Industrial Digital Twin Association e. V. Asset Administration Shell Reading Guide.

[11] Industrial Digital Twin Association e. V. Capability Description Submodel - IDTA Number: 02020.

[12] Industrial Digital Twin Association e. V. Specification of the Asset Administration Shell Part 1: Metamodel – IDTA Number: 01001-3-0-2.

[13] Industrial Digital Twin Association e. V. Specification of the Asset Administration Shell Part 2: Application Programming Interfaces – IDTA Number: 01002-3-0-3.

[14] B. Kelm, K. Balzereit, L. Moddemann, S. Myschik, and O. Niggemann. Application of a Model-based Reconfiguration Approach for the ISS COLUMBUS Environmental Control and Life Support System (ECLSS). In *33rd International Workshop on Principle of Diagnosis – DX 2022*, Sept. 2022.

[15] B. Kelm, K. Balzereit, S. Myschik, and O. Niggemann. Qualitative Graph-based Reconfiguration for Serially Constrained Hybrid Systems. In *34th International Workshop on Principles of Diagnosis (DX'23)*, Sept. 2023.

[16] M. Knitt, S. Plambeck, J. C. Wieck, J. Kohlisch, S. Balduin, E. M. Veith, J. Schyga, J. Hinckeldeyn, G. Fey, and J. Kreutzfeldt. Towards the automatic generation of models for prediction, monitoring, and testing of cyber-physical systems. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–4, 2023.

[17] L. Moddemann, H. Steude, A. Diedrich, and O. Niggemann. *Discret2Di - Deep Learning based Discretization for Model-based Diagnosis*. Nov. 2023.

[18] L. Moddemann, H. S. Steude, A. Diedrich, I. Pill, and O. Niggemann. Extracting knowledge using machine learning for anomaly detection and root-cause diagnosis. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2024.

[19] O. Niggemann, B. Stein, A. Vodencarevic, A. Maier, and H. Kleine Büning. Learning behavior models for hybrid timed systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1):1083–1090, Sep. 2021.

[20] O. Niggemann, B. Stein, A. Vodenčarević, A. Maier, and H. Kleine Büning. Learning Behavior Models for Hybrid Timed Systems. In *Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1083–1090, Toronto, Ontario, Canada, 2012.

[21] P. J. Overlöper, L. Moddemann, N. Hranisavljevic, A. Windmann, and O. Niggemann. Discretization of CPS Time Series with Neural Networks. In *IEEE International Conference on Emerging Technologies and Factory Automation*, Padova, 2024.

[22] S. Plambeck and G. Fey. Decision tree models of continuous systems. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022.

[23] S. Plambeck, M. Schmidt, A. Subias, L. Travé-Massuyès, and G. Fey. Usability of Symbolic Regression for Hybrid System Identification - System Classes and Parameters. In I. Pill, A. Natan, and F. Wotawa, editors, *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*, volume 125 of *Open Access Series in Informatics (OASIcs)*, pages 30:1–30:14, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[24] M. Redeker, D. Quirin, R. Schroeder, T. Klausmann, A. Löwen, A. Wollbrink, H. Stichweh, S. Althoff, A. Bender, W. Sextro, and M. Hesse. Towards a One-Stop-Shop Solution for the Application of Data-Driven Value-Adding Services in Production. In *IEEE ETFA*, 2024.

[25] H. S. Steude, C. Geier, L. Moddemann, M. Creutzenberg, J. Pfeifer, S. Turk, and O. Niggemann. End-to-end MLOps integration: a case study with ISS telemetry data. In *ML4CPS – Machine Learning for Cyber-Physical Systems*. Helmut-Schmidt-Universität Hamburg, 2024.

[26] A. Vignolles, E. Chanthery, and P. Ribot. Hybrid model learning for system health monitoring. *IFAC-PapersOnLine*, 55(6):7–14, 2022.