Flexibility of Modular and Accountable MLOps Pipelines for Cyber Physical Systems

Philipp Ruf, Christoph Reich
Institute for Data Science, Cloud Computing and Security *(IDACUS)*
Hochschule Furtwangen University (HFU)
Furtwangen, Germany
email: {Philipp.Ruf, Christoph.Reich}@hs-furtwangen.de

Djaffar Ould-Abdeslam
IRIMAS
Université de Haute-Alsace (UHA)
Mulhouse, France
email: djaffar.ould-abdeslam@uha.fr

***Abstract*—Operations within a Cyber Physical System (CPS) environment are naturally diverse and the resulting data sets include complex relations between sensors of the shopfloor devices setup, their configuration respectively. As Machine Learning (ML) can increase the success of industrial plants in a variety of cases, like smart controlling, intrusion detection or predictive maintenance, clarifying responsibilities and operations for the whole lifecycle supports evaluating the potentially feasible scenarios. In this work, the need for highly configurable and flexible modules is demonstrated by depicting the complex possibilities of extending simple Machine Learning Operations (MLOps) pipelines with additional data sources, e.g., sensors. In addition to the particular modules core functionality, arbitrary evaluation logic or data structure specific anomaly detection can be integrated into the pipeline. With the creation of audit-trails for all operational modules, automated reports can be generated for increasing the accountability of the different physical devices and the data related processing. The concept is evaluated in the context of the project Collaborative Smart Contracting Platform for digital value-added Networks (KOSMoS), where a sensor is part of an ML pipeline and audit trails are realized using Blockchain (BC) technology.**

***Keywords*—CPS; ML; MLOps; Deployment; Modularization.**

I. Introduction

In a fast evolving and interconnected world of user-specific needs, spontaneous demands on individually or rarely manufactured goods and the time span of completing such an order are new challenges to industrial operations. The fourth industrial revolution, known as *Industry 4.0* may be interpreted as the integration of interconnected systems and Internet of Things (IoT) in manufacturing [1]. Also known as CPS, this trend focuses on the deep integration of physical artifacts and informational entities [2], producing a huge amount of operational data. As research in this field is ongoing and concepts are refined continuously, topics like *cobots* (e.g., cooperating robots), further personalization, bio-economy, green computing and other sophisticated technology is summarized as *Industry 5.0* [3].

The usage of ML technology is always dependent on the theoretical feasibility of a respective scenario, the operational infrastructure, applied field devices and overall quality requirements. As production data originated within an organization's CPS plants, the dedicated combination of physical infrastructure and software is geared to the specific setup. While the field of ML emerges in the CPS domain, many Artificial Intelligence (AI)-driven use cases, as for example automated traffic signaling systems or Wireless Sensor Network (WSN) security and privacy enhancement, as outlined in [4], have been implemented in real-world scenarios. Such smart environments are also sometimes termed Artificial Intelligence of Things (AIoT) [5].

As the solutions to the respective problems are often depicted in detail, most literature lacks of comparable integration and management steps of devices in ML environments. It is more common to apply well-known data sets for demonstration and evaluation purposes. Another aspect of related work is executing ML operations on commercial infrastructure or services, as in [6]. In this context, an organization's data privacy policies are often threatened or cannot be met. Engineering an ML task results often in a static implementation and work is carried out within a dedicated environment, comprising specific hardware properties and libraries. Solving an ML problem is not an atomic task, but consists of a pipeline which can be interpreted as a domain-specific and integrated ML platform [7], containing various sub-processes.

When performing maintenance in a CPS plant, both the digital and physical shopfloor configuration must be tested extensively before production can continue. By serving a modular digital environment for CPS operations, an order-specific configuration of shopfloor devices can be dynamically deployed. The accountability and reliability of modules may impact the decision of pipeline compositions, too. In the case of on-demand manufacturing, many common tasks of the devices in a production-line can be automated.The whole pipeline must be held accountable, e.g., audit trails for each involved module operation must be persisted. When considering the obstacles around creating even simple ML pipelines [8], modularization of operations can improve stability and reliability when environmental circumstances change.

In the work on hand, the combination of CPS, ML and required quality aspects, is depicted by related work in Section II. By clarifying MLOps principles with respect to CPS in Section III, the foundation for discussing flexible ML pipelines in Section IV is given. The work is concluded in Section V.

 ISBN: 978-1-68558-017-9

II. Related Work and State-of-the-Art

As AI-based systems become more and more part of modern society, there are also public competitions backed and promoted by governments, e.g., as illustrated in [9]. In order to assure the participation and success of such events, up-to-date and user-friendly topics, as for example MLOps or AutoML, enable non-technical interested parties. A comprehensive overview of ML algorithms and their applications in real-world scenarios was given by Sarker in [10]. In outlining the most common algorithms functionality and intended usage, the importance of characteristics in data to be processed was accentuated. The different phases of MLOps and responsibilities of involved actors were clarified in [11]. With presenting a comparable list of supportive tools, an overview of ML-related environments, appropriate for different tasks was given. Although the work on hand describes an accountable and modular approach of defining MLOps pipelines, capable of implementing a variety of real-world setups, only a theoretical evaluation is performed and no involvement in public competitions took place.

A. Machine Learning for IoT and CPS

The utilization of ML techniques on data originating from industrial devices, e.g., mills, laser cutters, etc., has been implemented within many organisations so far. Sharma et ala [4] surveyed different efforts of ML with respect to IoT, e.g., different embedded devices and cloud-IoT platforms. Fei et al. [12] gave a comparative overview of ML-enabled data stream analytics. In the current literature for the most typical ML applications in CPS (smart grid, intelligent transport systems and smart manufacturing), various tasks and the respective ML techniques are depicted including the algorithms time complexity. In addition to basic ML techniques, incremental- and online learning is overviewed by the authors. In [3], use cases and further aspects of the *Industry 5.0* paradigm are clarified and an overview of technologies applied in the field of CPS is presented. In general, this is a refinement, and utilization of its predecessor, Industry 4.0, where interconnection of devices, humans and AI is extensively applied to industrial processes and scenarios. A containerized AIoT framework for enabling Continuous Integration, Continuous Delivery (CI/CD) of ML models and their deployment on highly configurable edge environments was shown by Raj et al. in [5]. When presenting an air-quality control system scenario in a distributed environment, e.g., conditions of different rooms, the model drift at the respective edges and a retraining with location-specific information was discussed. As outlined in [13], when developing a CPS operation, Digital Twin (DT)s are commonly applied in order to combine an abstraction of physical assets with the industrial application. Using a set of digital representations of a physical device, e.g., a module pipeline, production lines can be abstracted and actuated. Although there is an overlap with CPS operations, no holistic view on possible scenarios or specific deployment setups is considered in the work on hand. Rather, a bottom-up approach for modularization and deployment of modules using the KOSMoS framework is shown. As a management system and synchronization among DTs are some of the biggest challenges for the overall quality in a scenario [13], their composition of well defined and evaluated pipeline modules is one possible flexibility enhancement.

B. ML Quality and Deployment

The integration of trained ML models in preexisting logic is always application-dependent and, therefore, different quality requirements exist. For example, it may be required to deploy modules with consideration of certain restrictions or properties like scalability and serverless execution. The deployment of ML models as a nano-service was proposed by Paraskevoulakou et al. [14], where hardware resources were abstracted in order to provide a massive-scaleable *ML*-Function as a Service (FaaS), using the Apache OpenWhisk framework. Therefore, the same preprocessing pipeline of an offline-trained model is applied to unseen input data and forwarded via Representational State Transfer (REST) calls until the pre-trained model is invoked. Dependent on the technology stack of an operation or organization, such FaaS strategies may be integrated within the underlying infrastructure system itself. An overview and comparison of four open-source serveless platforms was given by Li et al. [15]. Mechanisms like the kubernets Horizontal Pod Autoscaler (HPA) automate resource-based scaling of pods by interpreting gathered metrics. On the other hand, stateful modules must be implemented with respect to such environmental conditions. A multi-target compiler for ML-model deployment was introduced in [16], where Predictive Model Markup Language (PMML)-compatible models are represented as a set of templates. These building blocks are applied in code generation for efficient production execution on single- and multi Central Processing Unit (CPU) and Graphics Processing Unit (GPU) systems. With respect to quality management systems, various real-world examples were outlined by Lee et al. [17], targeting the predictive maintenance in Industry 4.0. Amongst others, external data and multiple sensors represent the benefit of sensor fusion techniques. A comprehensive overview of quality dimensions, e.g., intrinsic, contextual, accessible and representational, with respect to *Big Data* was given in [18]. In addition to discussing data quality metrics for measuring the dimensions, quality scores are proposed for evaluation. A variety of quality attributes for microservice architectures is described in [19]. Architectural design decisions must be taken into consideration for classic requirements like the scalability or availability of an application, too. Aspects of a CPS demonstration cell were discussed in [20], where different devices simulated a production line and the generated data was used to predict the quality of a workpiece. While utilizing different ML techniques for the various CPS parts, there were challenges regarding the synchronizations and inconvenient labeling procedures. As the work on hand focuses on the deployment and interaction schemes of modularized ML pipeline parts, no attention is given to framework details or cross compilation. Due to the possible and likely fusion of datasets

 ISBN: 978-1-68558-017-9

originated from different sensors, difficulties can occur when synchronizing events. As the data and modules implementation quality is vital to the success of AIoT and CPS operations, domain-specific quality dimensions and requirements must be engineered beforehand.

III. Environments of ML Pipelines for CPS

A. ML in Hierarchical CPS Environments

The physical and digital setup and configuration of a shop floor is always specific to the CPS operations and architectures are often presented in a high-level manner. In literature, the commonality is often a top-down *1-to-n* connection of clouds, edges or fogs, CPS nodes, shopfloor devices and their integrated sensors and actuators respectively. Using such a basic depiction of an CPS constellation allows for clear and simplistic (re-)design of an operations digital infrastructure. Constant monitoring of the whole system on different levels enables a holistic view of current and historical operations [11]. As shown in Figure 1, the constellation of an organization's CPS and the placement of ML-related modules is driven by and dependent on domain- and scenario-specific expert knowledge. As illustrated, the depicted phases of a pipeline, which are clarified in the following, are impacted by their respective locations within a CPS. As the MLOps configuration is based on this CPS inventory, scenario-specific decisions which impact the stability of the whole workflow must be identified during project requirement engineering. With respect to the overall architecture, antipatterns, e.g., as in [19], and technical ML depts, e.g., as in [8], must be taken into consideration, too. In order to ensure usable datasets for model training, the data management phase aims for the fulfillment of different requirements, including domain specific evaluation on raw data. A a basic data quality analysis indicates the usability of a dataset version, where different techniques, processes and metrics exist for structured, e.g., text, or unstructured data like images [11]. Triggering and (re)configuring modules in an shopfloor environment, alarming for required maintenance or contribution to production pipeline management decisions are possible effects. When device-specific quality dimensions are identified, appropriate metrics, as outlined in [18], can be configured with scenario-specific attributes and indicate usable data and data to be reevaluated, respectively. There are many technical solutions to store versionable data, e.g., Data Version Control (DVC), problem-specific databases, distributed file systems or Copy-On-Write (COW) block devices, but dependent on the type of data, amount, frequency and usage of versions from different times, only a few of them will scale. Another aspect is the tenant-specific access of data, privacy concerns and the integration into existing digital environments. As the access to problem-specific data versions is not trivial, a high-level access or data stewardship support is appropriate for data-intense and dynamic pipelines. Operations in the preprocessing phase, e.g., labeling or generating appropriate features, can be automated to a certain extent, dependent on the underlying problem and involved sensors. The configuration of ML architectures optimal hyperparameters cannot be derived from operational data [21] and is problem-specific. The training processes are deployed to more or less potent hardware, possibly executed in parallel with respect to multiple versions or hyperparameter configurations. The metrics and model training runs may vary in requirements, as for example time constraints for the training, the models execution time or accuracy demands. With the definition of an applications access to the model, e.g., the model's inference, the problem-specific input data can be evaluated by modules from the training pipeline. A complete flow from sensing data to the usage of a resulting model is depicted and formalized by expert knowledge. As the environmental quality of each sensor, device, node, edge or cloud system is of relevance to the compliance with defined Key Performance Indicator (KPI)s, constant monitoring and quality assessment of the systems must be assured. With enough computation power, CPS nodes, edge devices or cloud environments are capable of data preprocessing tasks. As the training of ML models often requires Graphics Processing Unit (GPU) resources, more powerful edge devices and cloud environments are attractive locations. The model integration, inference respectively, can be enabled within CPS devices, nodes, edges or the cloud. Training a model on CPS devices data and applying the resulting ML implementation requires domain-specific knowledge of various shopfloor configurations in the first place. The dynamic deployment of ML tasks in CPS is based on the concatenation of containers, respective KPIs and an association with data structures of the actual shopfloor devices.

B. Generic composition of Modules

The declaration of an ML pipeline, e.g., a module composition, is comparable to creating a Directed Acyclic Graph (DAG) of predefined functions. In general, a module receives some kind of information, executes its dedicated processing and finally produces an output to be interpreted by its successor. Therefore, the actual module logic simply has to utilize such interfaces in order to serve as a composable and exchangeable part of the system. As a pipeline module tackles a specific problem, results can be treated as independent parts of the overall solution. When splitting the functionality of an application into an appropriate number of modules, their composition may enable more flexible, diverse and reliable operations. As every ML-related task fits best to a specific algorithm, as overviewed in [10] and [4], the granularity of a module is impacted by the respective scenario. Although a specific module implementation structure and its overall purpose depends on the problem on hand, common interfaces and communication patterns for interaction within the pipelines enhance the overall structure and configuration. As the design of the phases modules may require automated scaling, deployment strategies and applied technologies differ. Although autoscaling features are present in Kubernetes-based serverless platforms, as shown in [15], e.g., comparing *Nuclio, OpenFAAS, KNative* and *Kubeless*, the overall communication and management patterns must still be defined. When *interact-*

 ISBN: 978-1-68558-017-9

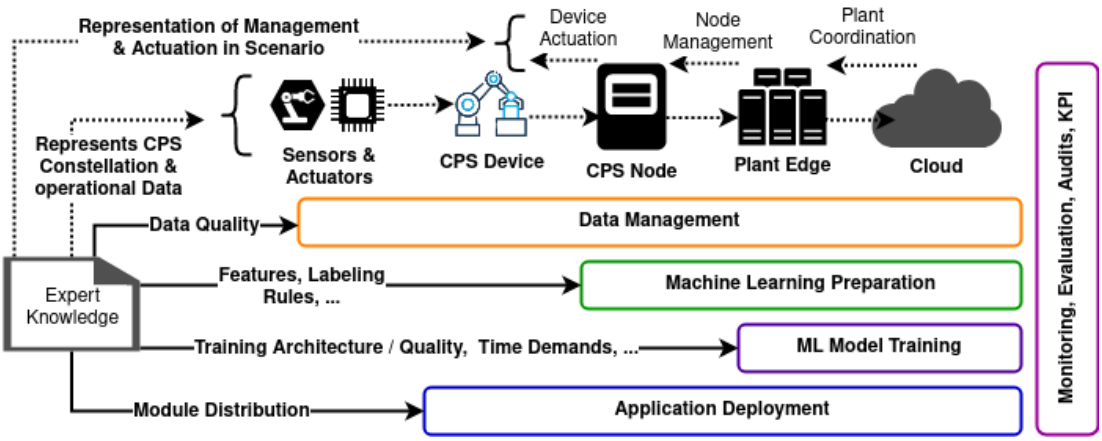

Figure 1. ML Tasks in CPS driven by formalized Expert Knowledge

ing with foreign systems, e.g., interfacing shopfloor devices or digital enterprise services, the respective module environment must be able to securely communicate with its counterparts in the first place. Especially, when a module is the origin of a pipeline, the data source is either produced by code, read from existing data structures or received via a foreign system by applying specific libraries. In addition to passive operations like reading a file from disk or pulling data from an endpoint, a module may be asynchronously triggered by a foreign system or an alien pipelines module respectively, or impact the environment by actuation. In contrast, a *module input* represents a connection to a specific storage, which enables the usage of previously persisted and task-related data. As the operations within a pipeline are often based on data science operations, common functionality like treating data to be persisted or read as *DataFrame* is viable. Such utilization will enhance convenience when applying the information within a subsequent module. When task-specific operations are designed in a generic way, whole modules may also be more flexible due to their parameterization. The *module output*, e.g., the persistence of module results in a task specific storage, can range from primitive datatypes to high-level objects, byte-code or trained models. By ensuring *module accountability*, each module instance can be audited separately. The logs can be of any kind and may also relate to custom operations for preceding modules in a specific pipeline setup. When consequently applying such logging patterns, the significance of task-specific audit-trails can be increased. Also, different parts of the *module evaluation* can be automatically executed when specified circumstances are met. They may strike when receiving data or as early as during their configuration with respect to fellow pipeline modules. Therefore, it is vital to the success of a module that metrics, parameters and custom tests are present for each implemented operation. Treating the various containers as independent *standalone* applications allows for implementing the whole learning pipeline as a dynamic and exchangeable configuration. The scheduling and deployment of tasks can also be carried out with respect to priorities and quality demands, as well as hardware requirements or modules in a pipeline. With executing the trained model version, an ecosystem of various monitoring hooks and mechanisms for assuring the specified KPIs is implied. Depending on the overall application goals, assertions of the declared ML tasks quality metrics influence the systems decision of automatically retaining the utilized model or performing actuation actions on the shop floor, respectively.

IV. Flexible Pipelines for ML in CPS

A. Exemplary Deployment and Accountability with KOSMoS

Depending on the complexity of data and the algorithms applied during processing, the optimal deployment technologies and techniques, the DAG of modules and the persistence of data varies. In the context of the KOSMoS project, a framework for the server-side management of client-side ecosystems was designed with respect to the CPS and shopfloor environments. By creating containers for each dedicated module, a communication protocol for interaction among them and defining a storage procedure, pipelines are depicted as JavaScript Object Notation (JSON) objects. As shown in Figure 2, a global platform (upper part) allows for the configuration of modules for pipelines which are applied in the digital environment of a system (bottom right part) and relates to a physical shopfloor (bottom left part). In such a KOSMoS setup, accountability of executed module operations, as well as significant device behaviour can be realized by providing pipeline- and tenant-specific access to the cloud-based BC technology. On the other hand, regular events can be transmitted to the BC as a hash, e.g., representing a interval of sensor data. Later on, the hashes can be compared to the respective data versions intervals at the pipelines local environments storage. There is a *n-to-n* relation between sensors and ML pipeline modules, as well as for modules among themselves. Basically, one or more sensors from one or more devices are configured with one or more pipeline modules. Therefore, sensor- and data-fusion is possible and likely to occur in CPS scenarios. When an applications infers a shopfloor-related AI-based model, automated actuation may occur at physical machines. As one promising area of CPS

 ISBN: 978-1-68558-017-9

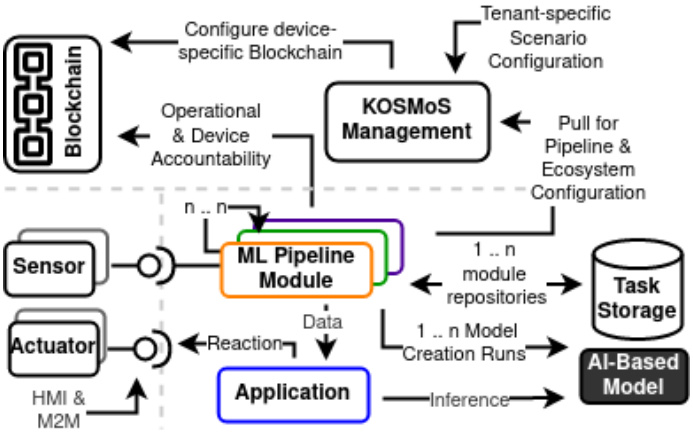

Figure 2. Accountable module Deployment with KOSMoS

is the human-robot coworking [3], the utilization of Human Machine Interface (HMI) can enhance productivity. When Machine to Machine (M2M) communication takes place on the shopfloor, authorized devices may reconfigure a production line's parameters, or interfere in dangerous situations. This is similar to another aspect of KOSMoS, where the possibility of cross-tenant cooperation is realized.

B. Simple Pipelines

In the following, a simple ML scenario is described, where the structure of each separate module comprises overall configuration parameters, as well as the interfaces described in Section III-B. In Figure 3, a temperature sensor is the origin of environmental data and the depicted ML pipeline implements the prediction of labels for the environment's next interval. Therein, every communication must be taken into consideration during evaluation of a module (e.g., Foreign System interaction, Configuration, input, Logging and Results). In the following, the assumption is that data quality requirements for this problem are known and an algorithm takes care of automated labeling of the sensors intervals. Therefore, the various preprocessing operations can be completely automated and a well-formalized *timeseries* data structure arises. Another aspect of this pipeline construction technique is the independence and dynamicity of module instances, enabling a high degree of interoperability and exchangeability of modules. The implementation of preprocessing device data and the model training will presumably differ for each type of ML task and requires the respective domain knowledge as well as software engineering and data-science capabilities. As implied in Figure 2, the digital shopfloor environment is capable of actuating the same devices from which data is already sensed. In controlling and reconfiguring physical surroundings, specific circumstances will require an adjustment of preexisting pipeline modules, too. The generic implementation of modules and asynchronous communication patterns allow for specific databases which are used for persisting a module's result and receiving the preceding outcomes, respectively. In the following, each module involved in the mentioned scenario is described.

a) Receive Sensor Temperature: In order to utilize CPS data, a reference to the respective devices has to be a declared for initially loading the correct data into the pipeline. When assuming a shopfloor device comprises primitive communication mechanisms, direct access to data via Universal Asynchronous Receiver / Transmitter (UART), Inter-Integrated Circuit (I2C), Serial BUS or other technologies is probable. As in the context of a CPS, industry standards for communication with such devices like *Simens S7* or Open Platform Communications Unified Architecture (OPC-UA) are more likely to occur and many libraries and communication models, e.g., pull vs. subscribe, exist. When deciding for a specific communication protocol, as OPC-UA was chosen for this example, various technology-specific parameters, e.g., how to connect to the specific machine in order to retrieve the required data, must be set. On the other hand, the received data and the data versions respectively, must be persisted within the task-specific storage and annotated with metadata related to this specific data version, e.g., module runtime, possible anomalies, and others. In addition to persisting task-specific operations, the logging of the operational context, e.g., the OPC-UA servers statistics for the respective interval, enhances the accountability and debugging of this specific module configuration.

b) Temperature Data Quality: When handling domain-specific hardware such as a temperature sensor, the respective datasheet most certainly clarifies circumstances in which the product works best. By performing checks for anticipated behavior of the dataset, basic data quality assessment can be carried out. In separating the domain-specific checks from well-formatted dataset versions, spontaneous exchanges or additional assertions related to scenario-specific quality assurance are made possible. Naturally, the access to previously sensed and persisted task-specific data, e.g., referencing the former modules output, must be configured in order to fulfill the processing of this module. In addition, by defining value ranges as well as other sensor-, scenario-, or domain-specific completeness indicators, a basic data quality assessment can be implemented. Additionally, techniques for *repairing* obvious outliers or anomalies in a dataset version may help in creating more reliable operations. In addition to creating an evaluated version of previously sensed data ready for preprocessing, various module-specific pieces of information, e.g., the justification of the data quality assessment, occurred anomalies, etc., allow for a more fine-grained monitoring, reporting and accountability.

c) Temperature Data Preprocessing: As the preprocessing of a dataset determines the ability of being used during ML model creation, the data structure originating from preceding modules must be interpretable, appropriate for the specific scenario, and allow for (semi)automated feature generation or engineering. When data cannot automatically be labeled and unsupervised learning is not an option, a *lambda*-like architectures can be applied. Therein, the base-knowledge, e.g., training-, test- and validation-dataset is extended with *new*

 ISBN: 978-1-68558-017-9

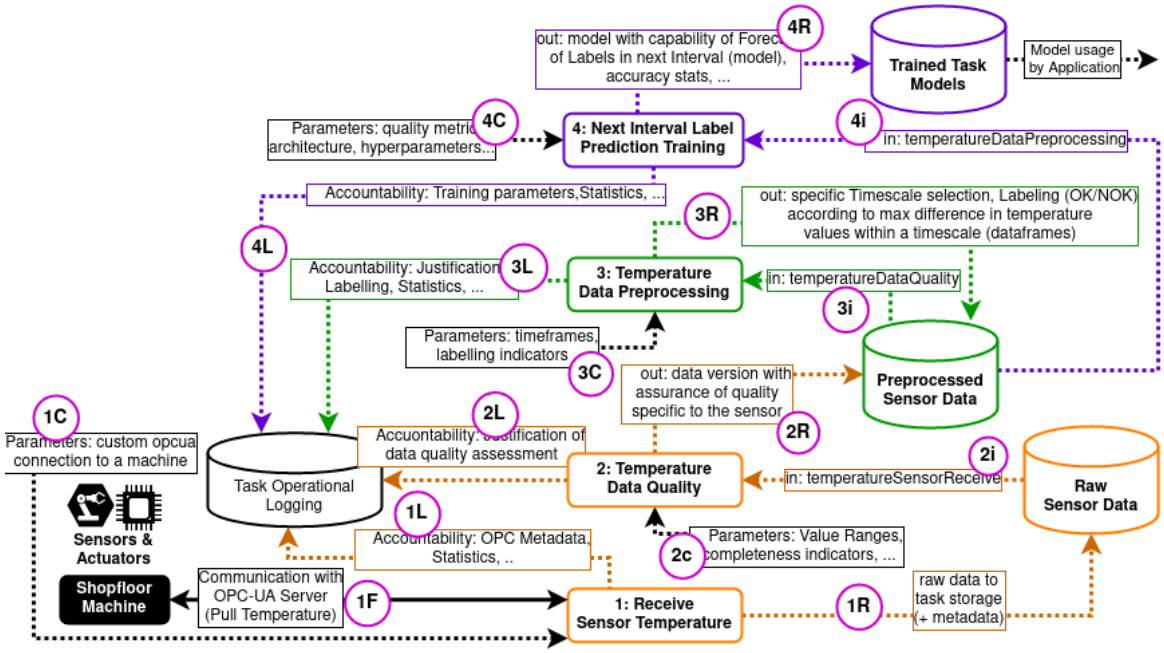

Figure 3. Module Pipeline Example for Temperature Label Prediction

sample-sets, whenever manual processing, e.g., boxing specific areas in images, is done. Either this event may trigger the re-training of a previously trained model or activate a transfer-learning process. A manual start of the learning process or periodically executed training runs are also viable options. The sensed temperature data is up to this point ensured to correspond to a certain pattern. Therefore, a simple labeling script which determines if a period is to be treated as *OK* or not, can be easily formalized and applied to the current base-knowledge. In addition to the persistence of the labeled timescale selections, module-internal statistics and justification of the labeling outcome can help in *up-to-date* decisions regarding a pipeline. For example, it can be asserted if a module's version, algorithm and the selected parameter are feasible for the utilization with a specific kind of data version.

d) Next Interval Label Prediction Training: Within this crucial step, the cleansed and ready-to-use dataframes are separated into test-, training- and validation-datasets and processed by the modules ML framework, e.g., generate the actual ML model. The feasibility within a scenario dependents on the module's implementation, its configuration of the learning architecture and feasible hyperparameters, as well as quality metrics for determining the model's performance with respect to the problem at hand. As the applied ML technologies and libraries impact the implementation complexity and required resources, this kind of module is most likely to be adjusted over time in order to reach the best performing outcome. This is also one of the modules where it is feasible to utilize *AutoML* functionality in parallel with hand-selected ML architectures and their hyperparameters, respectively. In addition to generating a model with the capability of forecasting labels for the next interval, the accuracy of a model and other technology-, custom-, domain- or quality-specific evaluation outcomes are versioned alongside.

e) Model Usage by Application: Once there is a suitable model available within the task-specific repository, its integration into the overall application takes place. According to whether the application itself is required to be recompiled for utilization of the new model or an additional independent lightweight execution environment module is created for its inference on demand, dynamic deployment of the up-to-date versions is possible. When referring to on-the-fly exchangeability, either the DevOps pipeline states how to dynamically use a model version or the application itself provides 'online'-configuration possibilities, e.g., changing a REST endpoint and other parameters. In addition to the model's specific ML-framework libraries, the logic for preprocessing or referencing the input parameters, e.g., data used for predictions, must be available to the module. When providing the module via REST, many additional parameters like listening ports, key material for transport encryption, allowed routes, e.g., calling modules and applications and other custom application-specific configurations must be defined. In addition to the model's usage and basic statistics, domain-specific information and inference results may be persisted in the dedicated task-specific repositories. Based on these events, it is possible to generate additional insights, or process information within a novel pipeline's modules.

C. Dynamic and Extensible Pipelines

As outlined previously, any generated data version can be applied to any module. By reusing pipelines up to a specific point, new scenarios, versions of scenarios or experiments are configurable with minimum effort, e.g., implementing a

 ISBN: 978-1-68558-017-9

new module while considering accountability guarantees. As imaginable, the possibilities and complexity for configuring an MLOps pipeline rise when there are multiple origins of data, e.g., available sensors. When an additional sensor becomes available on the shopfloor, the respective data management- and preprocessing-modules must be implemented while respecting the overall scenario's quality requirements [17] and demands on data quality [18]. In order to train a new model based on the combined sensor sources, existing workflows can be extended. This is similar to when a sensor is replaced by a different type, but proved parts of the workflow can be reused. On the other hand, a dedicated pipeline results in a sensor-specific model which can be combined with other existing models in the application phases. Also, attention should be payed to the many pitfalls of the sensor-data fusions data management and preprocessing phases, like scenario-specific requirements or timestamp synchronization, as in [20]. Dependent on the overall technology stack, existing management systems, as in [15], can be used as automated deployment manager, too. While the preexisting modules of a pipeline can transparently continue to process dedicated data versions and serving resulting information, additional pipelines, pipeline variants or module configurations can be defined for other scenarios. With ensuring the regular productive workflow and being able to experiment with potential improvements, the utilization of modules comprising Automated Machine Learning (AutoML) functionality becomes a promising aspect.

V. Conclusion

In this work, an approach of a flexible, module-based and accountability-enhanced pipeline definition for MLOps-conform implementation was described. In addition to clarify requirements to CPS-related module interactions, details on operations and persistence strategies were exemplary depicted and benefits of formalizing ML scenarios were highlighted. The different devices on a shopfloor can be rearranged, replaced or used in a novel way, which is why the possibility of dynamic updates to existing pipelines was briefly discussed. As the resulting dynamic pipelines may involve complex relations and dedicated meaning within a CPS, it is a challenge to provide accurate monitoring of each component. Although management frameworks for such distributed digital environments exist, hardware restrictions and threats to operations, e.g., bottlenecks or deadlocks, must be considered and modules should be adjusted accordingly. In future work, the integration of AutoML capabilities and an assertion of most feasible frameworks for different types of tasks are promising topics. Another aspect of using such a flexible structure is the possibility of evaluating quality attributes of pipeline compositions with respect to specific scenarios beforehand, due to using simulators for environments and device data.

Acknowledgement

This work is funded by the Federal Ministry of Education and Research (BMBF) under reference number 02P17D022 and supervised by Projektträger Karlsruhe (PTKA), Germany. The content was developed within the research project KOSMoS - (https://www.hs-furtwangen.de/en/research/forschungsprojekte/kosmos/).

References

[1] J. Lee, "Smart Factory Systems," *Informatik-Spektrum*, vol. 38, no. 3, pp. 230–235, jun 2015.

[2] S. Wang, J. Wan, D. Li, and C. Zhang, "Implementing smart factory of industrie 4.0: An outlook," *Int. J. Distrib. Sen. Netw.*, vol. 2016, pp. 7:7–7:7, Jan. 2016, accessed: 24.06.2022. [Online]. Available: https://doi.org/10.1155/2016/3159805

[3] B. Chander, S. Pal, D. De, and R. Buyya, "Artificial intelligence-based internet of things for industry 5.0," in *Artificial Intelligence-based Internet of Things Systems*. Springer, 2022, pp. 3–45.

[4] K. Sharma and R. Nandal, "A literature study on machine learning fusion with iot," in *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, 2019, pp. 1440–1445.

[5] E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, "Edge mlops: An automation framework for aiot applications," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 191–200.

[6] P. Singh, *Machine Learning Deployment Using Kubernetes*. Berkeley, CA: Apress, 2021, pp. 127–146, accessed: 24.06.2022. [Online]. Available: https://doi.org/10.1007/978-1-4842-6546-8_5

[7] H. E, K. Zhou, and M. Song, "Spark-based machine learning pipeline construction method," in *2019 International Conference on Machine Learning and Data Engineering (iCMLDE)*, 2019, pp. 1–6.

[8] e. a. Sculley, D, "Hidden technical debt in machine learning systems," *NIPS*, pp. 2494–2502, 01 2015.

[9] E. J. Maier, "Advancing artificial intelligence and machine learning in the us government through improved public competitions," *arXiv preprint arXiv:2112.01275*, 2021.

[10] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, pp. 1–21, 2021.

[11] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam, "Demystifying mlops and presenting a recipe for the selection of open-source tools," *Applied Sciences*, vol. 11, no. 19, 2021, accessed: 24.06.2022. [Online]. Available: https://www.mdpi.com/2076-3417/11/19/8861

[12] X. F. et al., "Cps data streams analytics based on machine learning for cloud and fog computing: A survey," *Future Generation Computer Systems*, vol. 90, pp. 435–450, 2019, accessed: 24.06.2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17330613

[13] M. W. Hoffmann, S. Malakuti, S. Grüner, S. Finster, J. Gebhardt, R. Tan, T. Schindler, and T. Gamer, "Developing industrial cps: A multi-disciplinary challenge," *Sensors*, vol. 21, no. 6, p. 1991, 2021.

[14] E. Paraskevoulakou and D. Kyriazis, "Leveraging the serverless paradigm for realizing machine learning pipelines across the edge-cloud continuum," in *2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2021, pp. 110–117.

[15] J. Li, S. G. Kulkarni, K. K. Ramakrishnan, and D. Li, "Analyzing open-source serverless platforms: Characteristics and performance," *CoRR*, vol. abs/2106.03601, 2021, accessed: 24.06.2022. [Online]. Available: https://arxiv.org/abs/2106.03601

[16] O. Castro-Lopez and I. F. Vega-Lopez, "Multi-target compiler for the deployment of machine learning models," in *2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 2019, pp. 280–281.

[17] S. M. Lee, D. Lee, and Y. S. Kim, "The quality management ecosystem for predictive maintenance in the industry 4.0 era," *International Journal of Quality Innovation*, vol. 5, no. 1, pp. 1–11, 2019.

[18] I. Taleb, M. A. Serhani, and R. Dssouli, "Big data quality: A survey," in *2018 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2018, pp. 166–173.

[19] T. Schirgi and E. Brenner, "Quality assurance for microservice architectures," in *2021 IEEE 12th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2021, pp. 76–80.

[20] P. e. a. Burggräf, "Predictive analytics in quality assurance for assembly processes: lessons learned from a case study at an industry 4.0 demonstration cell," *Procedia CIRP*, vol. 104, pp. 641–646, 2021.

[21] P. Janardhanan, "Project repositories for machine learning with tensorflow," *Procedia Computer Science*, vol. 171, pp. 188–196, 2020.

 ISBN: 978-1-68558-017-9