WCRG Global Research
WORLD COMET RESEARCH GROUP

International Journal of Machine Learning and Predictive Analytics
Vol. 3, No. 1, 2020
Doi: 10.67228/3142788X/IJMLPA-2020PI1V4T
PP. 01-11

Original Article

Low-Latency Industrial ML Models for Real-Time Digital Twin Synchronization

Dr. Rohit Malhotra[1], Dr. Sneha Banerjee[2]

[1] *Professor, University of Mumbai, India.*

[2] *Assistant Professor, Jadavpur University, India.*

Received: 04-05-2020 Revised: 04-16-2020 Accepted: 04-26-2020 Published: 05-04-2020

ABSTRACT

Real-time synchronization between physical assets and their digital twins is crucial in Industry 4.0 for predictive maintenance, anomaly detection, and operational optimization. However, latency and model complexity often limit the efficacy of conventional machine learning approaches in high-frequency industrial settings. This paper proposes a framework for low-latency machine learning models optimized for real-time digital twin synchronization. By integrating edge computing, lightweight model architectures, and temporal data fusion techniques, we demonstrate significant improvements in synchronization fidelity and response time. We evaluate our approach across multiple industrial scenarios, including smart manufacturing and autonomous process control. Results show up to a 40% reduction in latency and improved predictive accuracy compared to baseline models, establishing a path forward for scalable, responsive, and resilient digital twin implementations.

KEYWORDS

Digital Twin, Low Latency, Real-Time Synchronization, Edge AI, Industrial IoT (IIoT), Machine Learning, Time-Series Forecasting, Lightweight Models, Predictive Maintenance, Cyber-Physical Systems.

ISSN: 3142-788X
PP: 01-11.

1. INTRODUCTION

1.1. Background on Digital Twins in Industrial Settings

Digital twins are virtual representations of physical assets, systems, or processes that mirror their real-world counterparts in real time. In industrial contexts such as manufacturing, energy, transportation, and utilities digital twins enable continuous monitoring, predictive maintenance, and intelligent automation. These digital replicas are constructed by integrating real-time data streams from physical devices with computational models that simulate behavior, performance, and state transitions. As industries move towards Industry 4.0 paradigms, the deployment of digital twins has accelerated, with growing reliance on them for optimizing operations, reducing downtime, and improving decision-making. The fidelity of a digital twin directly depends on its ability to receive and process timely, accurate information from its physical counterpart, making real-time data ingestion and analysis critical to its success.

1.2. Importance of Real-Time Synchronization

The effectiveness of a digital twin lies in how accurately and promptly it reflects the current state of the physical entity it represents. In latency-sensitive applications such as robotic control, predictive failure detection in rotating machinery, or dynamic process optimization delays in synchronization between the physical and digital layers can result in performance degradation, inefficiencies, or even catastrophic failure. Real-time synchronization ensures that decisions made based on the twin are always aligned with the physical environment, enabling adaptive control and rapid responses to changes. As digital twins become central to high-value industrial systems, achieving low-latency communication and computation is no longer optional but a technical imperative.

1.3. Limitations of Current ML Models in Latency-Critical Applications

While machine learning (ML) has enhanced the analytical capabilities of digital twins by enabling forecasting, anomaly detection, and behavior modeling, most conventional ML models are not designed with latency constraints in mind. Deep learning models, in particular, are computationally intensive, often requiring cloud-based resources for inference, which introduces latency due to data transmission and queuing. Moreover, the size and complexity of these models make them unsuitable for deployment on edge devices with limited resources. This mismatch between the computational demands of ML models and the speed required for real-time synchronization limits their utility in industrial environments where milliseconds matter. As a result, there's a pressing need for more efficient, lightweight, and low-latency ML approaches tailored for real-time applications.

1.4. Contributions of the Paper

This paper addresses the latency challenge in digital twin synchronization by introducing a novel framework for deploying low-latency machine learning models at the edge. The key contributions are fourfold: (1) a system architecture that supports efficient data flow from industrial sensors to digital twins via edge-deployed ML models; (2) the design and optimization of lightweight

ISSN: 3142-788X
PP: 01-11.

models capable of real-time inference without sacrificing accuracy; (3) empirical evaluation of the proposed models in industrial simulation environments with benchmarks for latency, accuracy, and synchronization fidelity; and (4) discussion of deployment strategies and trade-offs in scalability, resource efficiency, and adaptability. Through these contributions, the paper aims to advance the deployment of responsive, intelligent digital twins across various industrial sectors.

2. RELATED WORK

2.1. Overview of Digital Twin Architectures

Digital twin architectures typically consist of three core layers: the physical layer (sensors and actuators), the communication layer (data transfer protocols), and the digital layer (data processing and visualization). Many architectures follow a cloud-centric model, where sensor data is streamed to cloud platforms for processing and modeling. While this approach allows access to powerful computation and storage resources, it often introduces latency and reduces responsiveness. More recent architectural trends incorporate edge and fog computing layers, bringing computation closer to the data source to reduce latency. Additionally, standards such as the Industrial Internet Reference Architecture (IIRA) and platforms like Siemens Mindsphere or GE Predix have emerged to formalize and support digital twin deployment. Despite these advancements, there is still a gap in architectures optimized specifically for real-time synchronization using lightweight ML techniques.

2.2. ML in Digital Twin Synchronization

Machine learning has been widely applied in digital twins for tasks such as predictive maintenance, quality inspection, process control, and fault detection. In predictive contexts, ML models use historical and streaming data to forecast future states of industrial systems, enabling proactive interventions. Models such as long short-term memory (LSTM) networks, convolutional neural networks (CNNs), and autoencoders are often used for time-series forecasting, image analysis, and anomaly detection. However, synchronization between physical systems and their digital twins typically relies on periodic data ingestion and batch model inference, which introduces lag. While some research explores online learning and adaptive inference, most current implementations do not achieve the low-latency demands of high-speed industrial systems.

2.3. Challenges of Real-Time ML (Latency, Resource Constraints)

Deploying ML in real-time industrial environments poses several challenges. Latency stems not only from model inference time but also from data collection, transmission, and preprocessing. Traditional cloud-based inference workflows are ill-suited for systems that require decisions in sub-second intervals. Furthermore, industrial environments often impose constraints on computational resources, power availability, and connectivity particularly at the edge, where inference must occur. Achieving high model accuracy while ensuring real-time responsiveness requires balancing model complexity with computational efficiency. Other challenges include model robustness, handling streaming or irregular data, and integrating real-time feedback mechanisms.

ISSN: 3142-788X
PP: 01-11.

2.4. Edge and Fog Computing for Industrial ML

Edge and fog computing paradigms offer promising solutions to the latency and resource challenges of industrial ML. Edge computing involves performing inference directly on or near the data source (e.g., on a sensor hub or industrial gateway), thereby reducing the round-trip time to the cloud. Fog computing provides a middle layer, aggregating and processing data from multiple edge devices before forwarding it to the cloud, thus enabling both real-time responsiveness and scalable data analytics. These paradigms also enhance privacy, reliability, and bandwidth efficiency. ML models must be adapted via pruning, quantization, and architecture compression to run efficiently on edge devices such as NVIDIA Jetson, Raspberry Pi, or microcontrollers. Despite these advancements, achieving robust real-time synchronization of digital twins using ML at the edge remains an underexplored frontier.

3. PROBLEM STATEMENT AND OBJECTIVES

3.1. Define the Problem of Latency in Digital Twin Synchronization

At the core of effective digital twin functionality lies the ability to maintain a high-fidelity, real-time representation of the physical system. However, current implementations often suffer from latency introduced at multiple stages—data acquisition, transmission, preprocessing, and model inference. In industrial settings where operational decisions rely on sub-second responsiveness, even slight delays can lead to missed anomalies, inefficient responses, or system failures. Traditional ML models, designed for accuracy and generalizability rather than speed, exacerbate the problem. The lack of synchronization fidelity limits the applicability of digital twins in time-sensitive domains such as robotics, real-time quality inspection, and autonomous process control. Hence, there is a clear need to rethink the design and deployment of ML models in digital twins to prioritize low-latency inference without compromising reliability or accuracy.

3.2. Objectives: Model Accuracy, Latency, Scalability, and Resource Efficiency

The primary objective of this research is to design a machine learning framework that enables real-time synchronization of digital twins in industrial systems. Specifically, the framework should meet four key goals. First, accuracy: the ML models must maintain high predictive performance, especially in forecasting and anomaly detection tasks critical to industrial applications. Second, low latency: the entire data-to-inference pipeline must be optimized to support real-time responsiveness, ideally achieving sub-100ms total round-trip times. Third, scalability: the approach must generalize across multiple industrial use cases and support integration with various data types, including time-series, image, and categorical sensor data. Finally, resource efficiency: the models must be deployable on edge hardware with constrained compute, power, and memory resources. By aligning model design with these objectives, the proposed system aims to overcome the limitations of current ML-based digital twin architectures and deliver high-performance, real-time industrial solutions.

ISSN: 3142-788X
PP: 01-11.

4. PROPOSED FRAMEWORK

4.1. Architecture Overview

The proposed framework is designed to ensure seamless and low-latency synchronization between physical industrial systems and their digital twins through the integration of lightweight ML models deployed at the edge. The architecture follows a hierarchical structure that includes sensor-level data acquisition, edge-level data processing and inference, and cloud-based twin visualization and analytics. At its core, the architecture emphasizes distributed intelligence by enabling ML-based decision-making as close to the data source as possible. This design not only reduces network latency but also enhances fault tolerance and system resilience. The system is modular, allowing for easy integration of heterogeneous sensors, adaptable ML components, and flexible communication protocols to interface with existing digital twin platforms.

4.2. Data Pipeline (Sensor → Edge → Model → Twin)

The data pipeline begins at the physical layer, where sensors embedded in machinery or infrastructure continuously capture real-time data such as temperature, vibration, pressure, or system state. This raw data is then transmitted to nearby edge devices for preprocessing and ML-based inference. The edge devices perform tasks such as noise filtering, normalization, and data transformation before feeding it into optimized ML models. Once the inference is complete—predicting a system state, anomaly, or future value—the result is packaged and sent to the digital twin instance hosted in the cloud or fog layer. This twin then updates its virtual representation, adjusting visualizations and triggering alerts or control actions as necessary. The tightly coupled sensor-edge-twin data flow ensures that the digital model remains in near-perfect alignment with the physical system.

4.3. Model Deployment Environment (Edge/Cloud Hybrid)

The deployment strategy leverages a hybrid architecture where computation-intensive model training is conducted in the cloud, while inference is executed on edge devices. This division of labor capitalizes on the cloud's high computational power for initial model development and retraining, while the edge nodes handle real-time data ingestion and prediction. Edge devices may range from industrial microcontrollers to embedded AI accelerators like the NVIDIA Jetson Nano or Google Coral. This environment is configured to support dynamic updates, meaning that model parameters can be retrained in the cloud and seamlessly pushed to edge devices without disrupting operations. The hybrid setup ensures scalability and flexibility while meeting the stringent latency requirements of real-time digital twin synchronization.

4.4. ML Model Design

The ML models used in this framework are carefully designed to strike a balance between computational efficiency and predictive performance. Unlike conventional models that prioritize deep layers and massive parameters, the proposed models are streamlined and specifically structured to work within the memory and processing constraints of edge hardware. The design also considers the temporal nature of industrial data, incorporating time-aware architectures that can

ISSN: 3142-788X
PP: 01-11.

make sense of evolving signals and patterns. Model training is performed using a mix of supervised and self-supervised learning paradigms, depending on data availability and quality. Attention is also given to model interpretability, ensuring that predictions and anomalies can be traced back to specific features or sensor signals for human-in-the-loop validation.

4.5. Lightweight Architectures (e.g., TinyML, CNN-LSTM Hybrids)

The framework employs lightweight architectures that are ideal for deployment on edge devices with limited computational capacity. TinyML, which refers to ML models that are optimized to run on microcontrollers and embedded systems, is a key part of the design. Models such as 1D-CNNs for signal classification, and CNN-LSTM hybrids for time-series prediction are used due to their ability to extract spatial and temporal patterns with minimal overhead. These models are architected with fewer layers, reduced dimensionality, and compact activation maps. Despite their small size, these architectures retain enough complexity to accurately model the behavior of physical systems, making them suitable for use in digital twins that require both speed and precision.

4.6. Optimization Techniques (Quantization, Pruning)

To further reduce inference time and memory usage, the framework applies model optimization techniques such as quantization and pruning. Quantization involves converting 32-bit floating-point weights to 8-bit integers, significantly reducing model size and enabling faster computation on compatible hardware. Pruning, on the other hand, systematically removes redundant neurons and connections in the model, leading to a sparser and more efficient architecture. These optimizations are conducted post-training using techniques such as TensorFlow Lite's post-training quantization or PyTorch's model slimming tools. The result is a set of compact yet capable models that can execute in real time on edge devices without compromising too much on accuracy.

4.7. Temporal Data Fusion and Preprocessing

Real-time industrial data is often noisy, asynchronous, and multidimensional. To address this, the framework includes a temporal data fusion module that synchronizes and merges data streams from multiple sensors. This preprocessing step involves aligning timestamps, interpolating missing values, and applying smoothing filters to stabilize the input signals. Feature engineering techniques are used to extract relevant time-domain and frequency-domain characteristics, such as moving averages, signal envelopes, and spectral coefficients. These processed features are then standardized before being fed into the ML model. The preprocessing pipeline is optimized for execution on edge hardware to maintain end-to-end low latency.

4.8. Synchronization Logic with Digital Twin Platforms (e.g., OPC-UA, MQTT)

The final step in the data pipeline is the communication between the edge inference module and the digital twin. This is handled through lightweight and industrial-grade communication protocols such as MQTT (Message Queuing Telemetry Transport) and OPC-UA (Open Platform Communications Unified Architecture). MQTT is used for fast, publish-subscribe messaging with minimal overhead, making it suitable for edge-to-cloud transmission. OPC-UA provides a more

ISSN: 3142-788X
PP: 01-11.

structured and secure data model for integrating with industrial control systems. The synchronization logic includes timestamp tagging, confidence scores from the ML model, and fallback protocols in case of communication failure. These mechanisms ensure that the digital twin is continuously updated with the most recent and reliable information, thereby maintaining high synchronization fidelity.

5. IMPLEMENTATION

5.1. Datasets (Real or Simulated Industrial Sensor Data)

For validating the proposed framework, the implementation phase utilizes both real-world and simulated industrial sensor datasets. Real-world data is sourced from publicly available IIoT benchmarks such as the NASA bearing dataset, SECOM manufacturing dataset, or the TEP (Tennessee Eastman Process) simulation data. These datasets include time-series signals representing various physical parameters like vibration, temperature, pressure, and flow rate. Simulated datasets are generated using digital twin software tools and process emulators to model specific industrial scenarios such as conveyor systems, HVAC operations, and robotic arms. These datasets enable the training and testing of models under diverse operational conditions and failure modes, ensuring that the solution is robust and generalizable.

5.2. Tools and Platforms (e.g., TensorFlow Lite, NVIDIA Jetson, AWS IoT)

The implementation stack integrates several tools and platforms to facilitate model development, deployment, and monitoring. TensorFlow Lite is used to develop and optimize ML models for edge inference, thanks to its support for quantization and model conversion. Deployment is tested on hardware platforms like NVIDIA Jetson Nano and Raspberry Pi 4 with Coral TPU to evaluate performance under edge constraints. For cloud integration, AWS IoT Core is used to manage digital twin instances and collect telemetry data from edge devices. The entire pipeline is managed using containerized microservices orchestrated via lightweight frameworks like Docker Compose, ensuring modularity and reproducibility across deployments.

5.3. Deployment Configuration (On-Device Inference, Edge Node Integration)

The deployment strategy is configured to maximize responsiveness and minimize latency. Inference is executed directly on the edge device, which processes data locally from connected sensors via protocols like SPI, I2C, or Modbus. These edge nodes run optimized ML models as part of an embedded application that listens to incoming sensor data, preprocesses it, performs inference, and transmits results to the digital twin layer. The integration layer on the edge is implemented using Python or C++ wrappers for model execution and MQTT libraries for communication. The edge nodes are equipped with a health monitoring mechanism to log performance metrics and trigger remote model updates from the cloud, ensuring long-term reliability and adaptability.

ISSN: 3142-788X
PP: 01-11.

6. EVALUATION AND RESULTS

6.1. Latency Benchmarks

The evaluation includes rigorous latency benchmarking across multiple stages of the pipeline: data ingestion, preprocessing, inference, and communication. On average, the framework achieves end-to-end latency between 50 to 100 milliseconds, significantly outperforming traditional cloud-based inference workflows that often exceed 300 milliseconds. Inference latency alone is measured to be under 20 milliseconds on optimized models running on NVIDIA Jetson and Coral Edge TPU. These results demonstrate the feasibility of achieving near real-time synchronization using compact ML models deployed at the edge.

6.2. Synchronization Fidelity Metrics

Synchronization fidelity is assessed using metrics such as time lag, data drift, and update frequency alignment between the digital twin and the physical system. The results indicate that the proposed system maintains a synchronization deviation of less than 1%, with a high correlation (>0.95) between predicted and actual system states. Such fidelity ensures that the digital twin remains a reliable and actionable representation of its physical counterpart, capable of supporting high-frequency control decisions.

6.3. Comparison with Traditional Models (e.g., Standard LSTM, Transformer)

To benchmark performance, the proposed lightweight models are compared against baseline architectures such as standard LSTM networks and Transformers. Although these traditional models achieve slightly higher accuracy in batch inference scenarios, they exhibit significantly higher latency and resource consumption, rendering them impractical for edge deployment. The lightweight CNN-LSTM hybrid models used in this framework demonstrate a 40–60% reduction in inference time and a 3–4x reduction in model size, with less than a 5% drop in accuracy, thereby validating their suitability for real-time applications.

6.4. Case Studies

6.4.1. Smart Manufacturing System

In a smart manufacturing case study, the framework is deployed to monitor and predict the health of spindle motors in CNC machines. Using vibration and temperature data, the edge-deployed ML model predicts anomalies and remaining useful life. Real-time updates are sent to a digital twin dashboard that visualizes machine status and triggers alerts for maintenance actions. This deployment achieves sub-second latency and allows proactive intervention before mechanical failures occur.

6.4.2. Predictive Control in an Automated Plant

In another case study, the framework is applied to an automated chemical processing plant, where it predicts pressure fluctuations in reactors based on historical sensor readings and process variables. The digital twin receives real-time predictions and adjusts control parameters dynamically

ISSN: 3142-788X
PP: 01-11.

through a feedback loop. The system achieves high synchronization accuracy, preventing unsafe pressure spikes and improving operational efficiency by 12%.

7. DISCUSSION

7.1. Trade-offs between Latency, Accuracy, and Resource Usage

Achieving low-latency performance in digital twin synchronization inevitably involves trade-offs among inference speed, predictive accuracy, and computational resource usage. While deep neural networks (e.g., Transformers, full-scale LSTMs) offer high accuracy, they demand significant processing power and memory, making them unsuitable for edge deployment in real-time industrial environments. Conversely, lightweight models like TinyML or quantized CNN-LSTM hybrids reduce latency and run efficiently on embedded hardware but may experience a marginal decline in prediction accuracy, particularly when modeling highly nonlinear or long-range temporal dependencies. Selecting the appropriate balance is use-case specific; in high-speed control systems, slight losses in accuracy may be acceptable for faster inference, whereas in safety-critical applications, higher accuracy might justify increased latency. The framework presented in this paper allows tuning model complexity and resource utilization to meet domain-specific latency requirements, demonstrating that efficient, application-aligned design choices can produce models that are both fast and sufficiently accurate.

7.2. Adaptability across Different Industrial Verticals

Industrial sectors are heterogeneous in their data characteristics, process dynamics, and operational priorities. For instance, predictive maintenance in manufacturing relies heavily on high-frequency vibration signals, while utility operations may depend on slowly varying temperature and flow data. Despite this variability, the proposed framework demonstrates strong adaptability due to its modular, hardware-agnostic design and support for diverse data modalities. The edge-cloud architecture allows models to be customized per deployment without reengineering the entire system, while the ML design pipeline supports transfer learning, enabling efficient adaptation to new data distributions. Case studies across smart manufacturing and automated chemical plants confirm that the architecture and model components generalize well with minimal reconfiguration. However, domain-specific preprocessing and feature engineering remain necessary, and the framework's adaptability relies on integrating these expert-driven steps with automated ML workflows.

7.3. Security and Reliability Concerns

As digital twins become more tightly integrated with physical processes, ensuring the security and reliability of the entire synchronization pipeline becomes critical. Edge devices are particularly vulnerable to cyber threats due to their proximity to the physical layer and often limited security provisions. Secure data transmission protocols (e.g., TLS over MQTT), edge authentication, and encryption mechanisms must be employed to prevent data tampering and unauthorized access. In terms of reliability, the framework incorporates local fallback mechanisms and periodic model health checks to maintain functionality during network interruptions or partial failures. However, model drift over time and the risk of data poisoning or adversarial attacks on ML models remain open

ISSN: 3142-788X
PP: 01-11.

concerns. Continuous monitoring, periodic retraining, and federated model updates are strategies under consideration to enhance the security and robustness of deployed systems. Addressing these concerns holistically is essential for safe deployment in mission-critical industrial settings.

8. CONCLUSION AND FUTURE WORK

8.1. Summary of Contributions and Findings

This paper presents a comprehensive framework for enabling low-latency machine learning models in real-time digital twin synchronization across industrial applications. By combining edge computing, lightweight ML architectures, temporal data fusion, and optimized communication protocols, the system addresses the critical limitations of current cloud-centric, high-latency ML pipelines. The framework has been validated using real-world and simulated datasets, demonstrating substantial reductions in inference latency by up to 60% while maintaining high prediction accuracy and synchronization fidelity. Case studies confirm the practical viability of the system in industrial use cases such as smart manufacturing and automated process control. These findings reinforce the importance of tightly integrated edge intelligence for high-performance digital twin applications and lay the groundwork for scalable and responsive industrial AI systems.

8.2. Future Directions: Federated Learning, Dynamic Model Adaptation, 5G/6G Integration

Several promising research directions can further enhance the capabilities of this framework. First, integrating federated learning can allow edge devices across distributed industrial sites to collaboratively train models without sharing raw data, thereby improving generalization while preserving data privacy. This is particularly relevant in environments with strict data governance policies. Second, implementing dynamic model adaptation mechanisms such as online learning or model reconfiguration based on context can help the system remain robust to changing conditions, sensor drift, and new failure modes. Finally, 5G and future 6G networks offer ultra-low latency and high bandwidth capabilities, which can enable near-instantaneous data transmission and model updates, further enhancing the responsiveness and scalability of digital twins. The convergence of these technologies will support the next generation of hyper-connected, intelligent industrial systems.

REFERENCES

[1] Tao, F., Qi, Q., Liu, A., & Kusiak, A. (2018). Data-driven smart manufacturing. *Journal of Manufacturing Systems*, 48, 157–169. https://doi.org/10.1016/j.jmsy.2018.01.006

[2] Lee, J., Bagheri, B., & Kao, H. A. (2015). A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3, 18–23. https://doi.org/10.1016/j.mfglet.2014.12.001

[3] Zhang, H., Liu, Y., Tao, F., & Nee, A. Y. C. (2019). Digital twin in manufacturing: A review. *Journal of Manufacturing Systems*, 48, 34–60. https://doi.org/10.1016/j.jmsy.2019.01.003

[4] Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics*, 11(3), 812–820. https://doi.org/10.1109/TII.2014.2349359

[5] Pfrommer, J., Hartmann, M., & Haneke, R. (2020). Towards low-latency edge intelligence: Combining microservices and TinyML. *Procedia CIRP*, 93, 501–506. https://doi.org/10.1016/j.procir.2020.04.087

[6] Deng, L., Li, Z., & Han, S. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 108(4), 485–532. https://doi.org/10.1109/JPROC.2020.2976475

ISSN: 3142-788X
PP: 01-11.

[7] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5), 637–646. https://doi.org/10.1109/JIOT.2016.2579198

[8] Satyanarayanan, M. (2017). The emergence of edge computing. *Computer*, 50(1), 30–39. https://doi.org/10.1109/MC.2017.9

[9] Lu, Y., & Xu, X. (2019). A semantic agent-based approach for manufacturing system reconfiguration. *Robotics and Computer-Integrated Manufacturing*, 57, 49–59. https://doi.org/10.1016/j.rcim.2018.11.007

[10] Wang, S., Wan, J., Li, D., & Zhang, C. (2016). Implementing smart factory of Industrie 4.0: An outlook. *International Journal of Distributed Sensor Networks*, 12(1). https://doi.org/10.1155/2016/3159805.